UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA
Departamento de Ingeniería del Software e Inteligencia Artificial

TESIS DOCTORAL

Desarrollo dirigido por modelos para la simulación de tráfico rodado basada en agentes

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Alberto Fernández Isabel

Director

Rubén Fuentes Fernández

Madrid, 2016

Desarrollo dirigido por modelos para la simulación de tráfico rodado basada en agentes

Memoria que presenta para optar al grado de
Doctor en Informática

Alberto Fernández Isabel
Dirigida por el doctor:
Rubén Fuentes Fernández

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid
Madrid, Octubre de 2015

Agradecimientos

Esta tesis ha sido un desafío y una motivación a lo largo de todos estos años. Un objetivo inicialmente planteado cuando finalicé la Ingeniería Técnica en Informática de Sistemas en Aranjuez y que parecía inalcanzable en aquel momento. Tan lejos en esos comienzos, tan próximo cuando se empezaba a ver el final, pero siempre pidiendo un esfuerzo más, un minuto más de sacrificio. Pero nada de eso importa ahora. Para mí desde aquel momento, sentirme doctor era sentirme realizado a nivel profesional y aquí estoy.

El camino no ha sido nada fácil. Comencé el desarrollo de mi investigación mientras trabajaba en la Universidad Complutense de Madrid (UCM) como becario de colaboración. Aquel tiempo fue realmente duro, pues no podía continuar mucho más en esa situación laboral y sólo recibía denegaciones en las solicitudes de becas de investigación que presentaba. Estando a punto de replantearme si podía continuar o no con la tesis, apareció el Consejo Superior de Investigaciones Científicas (CSIC) para arrojar algo de luz sobre mi futuro. El trabajo que me ofreció era motivador, dándome la posibilidad de continuar con la elaboración del proyecto y a la vez ampliar mi experiencia laboral. Por ello agradezco al destino, a Dios, o simplemente a la casualidad, haber tenido esa suerte. Pero ante todo, hablar del CSIC es hablar de gente que me ha apoyado y ayudado, permitiéndome incluso utilizar los no pocos recursos de los que este organismo dispone. Por lo tanto, qué menos que darles las gracias a todos ellos en este momento.

En lo profesional y también en lo personal, quiero agradecer ante todo a mi director de tesis Rubén haber sufrido conmigo todo ese proceso. Desde aquí no me queda más que darle las gracias por su paciencia, sabiendo guiarme y motivarme en los momentos difíciles, peleando conmigo (pues él sabe bien que soy de cabeza dura), enseñándome a ser conciso en las explicaciones, e incluso poniendo ese punto de descontrol a veces tan necesario en las discusiones sobre cómo realizar una cosa u otra.

Recordar también a los colaboradores de mis artículos, por haber realizado el esfuerzo de contribuir y revisar el trabajo una y otra vez. Aquí quiero recordar en especial a María Guijarro, Gonzalo Pajares y Diego Blanco.

También quiero dar las gracias a Juan Pavón, por sus recomendaciones iniciales y por sus consejos específicos. Es bueno tener alguien que te da las primeras directrices y claves para comenzar tu investigación.

No olvido a todos los miembros del GRupo de Investigación en Aplicaciones Sociales e Interdisciplinares basadas en Agentes (GRASIA). Os agradezco enormemente vuestra información y crítica, y el interés mostrado en mí.

Aquí debo hacer mención a los proyectos de GRASIA y sus correspondientes financiaciones que han ayudado a la consecución de esta tesis: El proyecto ‘Social

Ambient Assisting Living - Methods (SociAAL)' (TIN2011-28335-C02-01) proporcionado por el Ministerio de Economía y Competitividad (MINECO), el programa de investigación 'MOSI-AGIL-CM' (S2013/ICE-3019) otorgado por la Comunidad Autónoma de Madrid and cofinanciado por el Fondo Social Europeo (FSE) y el Fondo Europeo de Desarrollo Regional (FEDER), y el 'Programa de Creación y Consolidación de Grupos de Investigación' (UCM-BSCH GR35/10-A). Finalmente agradecer el soporte procurado por la 'Red Científico-Tecnológica en Ciencias de los Servicios' (TIN2011-15497-E).

En cuanto a mi círculo más íntimo, debo dar las gracias en primer lugar a mi familia, a mi padre Jesús y a mi madre María Isabel, por todo vuestro incondicional apoyo. Sé lo difícil que es lidiar conmigo muchas veces, y os agradezco el esfuerzo mostrado desde que vine a este mundo hasta el día de hoy. Habéis conseguido que dé otro paso más en este viaje que es la vida.

Mención especial merece mi compañera en la vida y pareja, Leticia, la cual ha tenido que soportar todas mis variadas excusas debidas a la realización de este proyecto de investigación. Ella ha sabido entenderme, motivarme y animarme, viendo siempre lo mejor de mí y haciéndome saber que su hombro estaba ahí, a mi entera disposición.

Y finalmente, también mencionar a los amigos que han estado ahí preguntándome e interesándose por el largo proceso. Sin vosotros no sería quien soy actualmente.

El día que me siente al lado de Dios al final de mi vida,
espero no haber dejado ni uno solo de mis talentos por usar
y poder decirle
“he usado todo lo que me diste”.
–Erma Bombec

Resumen

El tráfico rodado es un fenómeno importante en las sociedades modernas, de ahí la relevancia de su estudio. La complejidad de las relaciones entre los individuos y otros objetos involucrados en él, junto a la cantidad y variedad de variables y parámetros que le afectan, hacen que su análisis en un entorno real sea muy complejo y costoso.

Las simulaciones de tráfico son una herramienta para abordar este problema. Permiten reducir la cantidad de datos reales a considerar y simplificar la complejidad de ciertas operaciones, y ofrecen un entorno controlado donde realizar los experimentos. Sin embargo, las simulaciones tienen también sus inconvenientes. Los modelos más genéricos requieren simplificaciones muy importantes que los alejan de la realidad plausible, centrándose en muy pocas características. Los modelos que consideran más factores deben ser muy específicos para conseguir mantener tratable su especificación, análisis y complejidad computacional. Además, el desarrollo de estas simulaciones suele requerir expertos con diferentes perfiles (por ejemplo, en tráfico y en las plataformas de simulación). Esta heterogeneidad causa frecuentemente problemas de comunicación y comprensión en los equipos de desarrollo.

La Ingeniería del Software Dirigida por Modelos (ISDM) ha sido propuesta para desarrollar simulaciones reduciendo esas desventajas. Se trata de una aproximación al desarrollo centrada en la elaboración de modelos y sus transformaciones semi-automatizadas, donde prácticamente toda la información empleada se hace explícita. En general usa lenguajes de modelado gráficos especializados para los diferentes roles participantes en los equipos. Esto permite representar las especificaciones intuitivamente, y facilita modularizar el desarrollo entre los distintos perfiles y relacionar sus contribuciones.

Esta tesis presenta un marco de desarrollo de ISDM para simulaciones de tráfico. El marco consta de tres componentes fundamentales: un lenguaje de modelado específico del dominio, herramientas de soporte, y guías para usar los anteriores.

El elemento clave es el Lenguaje de Modelado para el Tráfico (LMT). Éste se centra en los aspectos que influyen en el comportamiento de los individuos, junto con su evaluación de alternativas y toma de decisiones. La definición de estos elementos se basa en teorías de amplio uso en los estudios sobre tráfico, como el modelo Conductor-Vehículo-Entorno (CVE). Su especificación adopta propuestas del Modelado Basado en Agentes (MBA) y la Ingeniería del Software Orientada a Agentes (ISOA), tales como el uso del *agente* como elemento central del modelado, con un ciclo de percepción, razonamiento y actuación, y la representación explícita de sus objetivos.

Los expertos cuentan con dos herramientas principales de desarrollo. Un editor gráfico de especificaciones proporciona un entorno para diseñar las instancias de los modelos y validar con el LMT las estructuras obtenidas. Un generador de código guía a los usuarios en la transformación de las especificaciones a código fuente mediante asistentes semi-automatizados.

Además de las herramientas anteriores, se han utilizado en los casos de estudio diversas plataformas de simulación de tráfico, incluyendo MATSim, SUMO, y una específicamente desarrollada para esta tesis. El objetivo de utilizar múltiples plataformas ha sido validar la versatilidad de la solución presentada.

El último componente del marco de desarrollo son las guías para su uso. Estas sintetizan la experiencia de desarrollo con el marco para proporcionar consejo a los desarrolladores en la forma de utilizarlo.

Como se indicó anteriormente, la propuesta ha sido validada a través de varios casos de estudio. En ellos se han reproducido experimentos descritos en la literatura usando el marco de desarrollo propuesto. Los resultados muestran que la propuesta permite capturar toda la información pertinente para el desarrollo y facilita realizar modificaciones de las simulaciones a través de los modelos y las transformaciones. Esto mejora la reusabilidad de los artefactos utilizados, reduciendo los costes de desarrollo y mejorando la autonomía de los expertos.

Palabras clave: marco de desarrollo; lenguaje de modelado de tráfico; generador de código; guía de desarrollo; ingeniería del software dirigida por modelos; modelado basado en agentes; simulación de tráfico.

Abstract

Road traffic is a very important phenomenon in modern societies, which makes its study highly relevant in multiple contexts. The complexity of the relationships among the people and other objects involved in it, along with the number and heterogeneity of variables and parameters that affect it, renders their analysis in real settings complex and costly.

Traffic simulations appear as a tool to address the previous problems. They allow reducing the amount of actual data to consider, simplifying the complexity of certain operations, and presenting a controlled environment to carry out experimentation. However, simulations also present problems. The most general models make very important simplifications that consider only a few features, which produces concerns about their representation of a plausible reality. Models that consider more aspects need to be very specific in order to keep achievable their specification and study, and limit their computational complexity. Besides this, the development of these simulations usually demands experts from different fields (e.g. in traffic and simulation platforms). This heterogeneity frequently causes problems of communication and misunderstandings in development teams.

Model-Driven Engineering (MDE) has been proposed to develop simulations while mitigating these problems. It is a general software development approach focused on the specification of models and their semi-automated transformations. In it, almost all the information used in the development becomes explicit. In general, MDE uses graphical modelling languages tailored for the different roles participating in development teams. This allows an intuitive representation of specifications, and facilitates communication and modularization of work among roles, and relating the different contributions.

This work introduces a MDE framework for the development of traffic simulations. This framework includes three main components: a domain-specific modelling language, support tools, and guidelines to use the previous elements.

The main element is the Traffic Modelling Language (TML). This language is focused on the persons that participate in traffic through the elements and factors that affect them and their perception and decision-making. The definition of these elements is based on theories widely applied in traffic studies, like the Driver-Vehicle-Environment (DVE) model. Its specification adopts elements from Agent-Based Modelling (ABM) and Agent-Oriented Software Engineering (AOSE), such as using the *agent* as the core concept when modelling, with a perception, evaluation, and action cycle, and an explicit representation of its goals.

The framework supports experts with two main tools. A graphical model editor for specifications provides an environment to design the model instances and validate their structures with the TML. A code generator guides and supports users with semi-automated wizards to produce the transformations that generate the target source code from models.

Besides these tools, the case studies used several simulation platforms, including MATSim, SUMO, and one developed as part of the framework. The goal was validating the proposed solution in settings with different features.

The last component of the framework are the guidelines for its application. These synthesize the experience gained in developments with the framework to provide developers with guidance and hints on its use.

As said before, the simulation framework has been validated with several case studies. These have reproduced experiments already done in literature but using the proposed framework. Results show that the framework allows eliciting all the relevant information and facilitates the modification of simulations through their models and transformations. This improves the reusability of development artefacts, reducing the development costs and improving experts' autonomy in the development of simulations.

Keywords: development framework; traffic modelling language; code generator; development guidelines; model-driven engineering; agent-based modelling; traffic simulation.

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

Este capítulo describe la motivación que ha conducido a abordar esta tesis, así como los objetivos y el método de trabajo seguido en su realización. El capítulo finaliza describiendo la estructura y el contenido de este documento.

1.1 Contexto

El estudio del tráfico rodado en entornos reales es un proceso muy complejo. Necesita considerar una gran cantidad de variables y numerosos elementos participando en múltiples interacciones, que a su vez pueden producirse en distintos escenarios. Entre esos elementos se incluyen fundamentalmente personas, lo que implica consideraciones adicionales de salud y seguridad en los experimentos. Por ello, las simulaciones aparecen como un instrumento clave para estudiar el tráfico, ya que permiten simplificar la complejidad de los entornos considerados y controlar las variables implicadas.

El uso de simulaciones no está exento de problemas [2]. En primer lugar, existen limitaciones en la complejidad abordable de los modelos y sus ejecuciones, tanto desde el punto de vista de su comprensión como de su desarrollo y coste computacional. Ello lleva frecuentemente a tener que adoptar un compromiso entre simulaciones muy genéricas (que simplifican posiblemente en exceso la realidad que estudian) o muy específicas (para poder estudiar con más detalle los aspectos considerados). En segundo lugar, surge la necesidad de formar equipos de desarrollo heterogéneos (es decir, con participantes de diversos perfiles) cuando la simulación no es muy sencilla. Ello se debe a que se necesita tratar múltiples aspectos diferentes en el desarrollo, desde los específicos del dominio a los más técnicos. En este tipo de equipos surgen dificultades de comunicación (a causa de los malentendidos y la información implícita que se maneja entre algunos grupos de expertos con una base común), y se presentan diferencias en los objetivos de los participantes. En consecuencia, elaborar consensuadamente modelos de simulación compatibles a diferentes niveles de abstracción (por ejemplo, la teoría social y el diseño del código) se torna una tarea muy compleja. Por ello, no se puede garantizar que las simulaciones resultantes reflejen fielmente los modelos abstractos iniciales.

Por otra parte, el desarrollo de simulaciones de tráfico se hace actualmente siguiendo aproximaciones tradicionales centradas en el código. Esto quiere decir que los expertos en tráfico proporcionan modelos abstractos (muchas veces descritos en lenguaje natural) a los desarrolladores, que estos traducen manualmente a código e integran en la plataforma de destino. En ocasiones se pueden emplear especificaciones intermedias (por ejemplo con lenguajes de modelado o ecuaciones), pero generalmente

se usan con carácter de documentación. Esta forma de trabajar incrementa las posibilidades de cometer errores que pasen inadvertidos en los pasos sucesivos. Además, dificulta modificar las simulaciones, por ejemplo para introducir nuevas teorías, considerar otros aspectos del problema o cambiar la plataforma de simulación.

El uso de enfoques dirigidos por modelos ha sido propuesto para abordar este tipo de problemas. La Ingeniería del Software Dirigida por Modelos (ISDM, *Model-Driven Engineering*, MDE por sus siglas en inglés) [22, 38] organiza los proyectos de desarrollo en torno a *modelos*. Estos deben ser conformes con las especificaciones de Lenguajes de Modelado (LM) bien definidos, esto es, con una definición formal de al menos su sintaxis. El otro elemento clave de estas propuestas son las *transformaciones* (automáticas o semiautomáticas). Éstas suelen tomar como entrada modelos, y generar otros modelos (o añadir y modificar información en los ya existentes) o texto (por ejemplo, código fuente o documentación). De esta manera, el proceso de desarrollo es iterativo, en un ciclo continuo de refinado de modelos abstractos a otros más específicos, hasta que la generación de código fuente es posible.

Para las simulaciones de tráfico, una aproximación de ISDM implica el desarrollo de LM para especificar la simulación desde diferentes perspectivas (por ejemplo, el comportamiento del conductor, el funcionamiento del coche, y el entorno), a diferentes niveles de abstracción (por ejemplo, los del experto en tráfico y el programador), y considerando los diferentes roles que pueden tomar los individuos en el fenómeno (por ejemplo, conductores, peatones o pasajeros). Para la definición y uso de estos LM, los desarrolladores deben proporcionar editores gráficos para sus especificaciones y herramientas para transformarlas. También es preciso disponer de guías de desarrollo que indiquen la forma apropiada de construir estas especificaciones y refinarlas, organizar el desarrollo o manejar las herramientas.

El principal obstáculo que presenta el uso de este tipo de desarrollos es el esfuerzo inicial para crear la infraestructura de ISDM, ya que es mayor que el que se necesita para desarrollar manualmente una sola simulación centrándose en el código fuente. Este esfuerzo es compensado por la mayor reutilización de los artefactos del desarrollo de simulaciones (por ejemplo, especificaciones y transformaciones), y la descripción explícita de toda la información utilizada para crear estos productos [23]. Los LM, especificaciones y transformaciones pueden ser reutilizados, probados y definidos de forma incremental en diferentes proyectos. Por ejemplo, un LM para entornos urbanos es aplicable a diferentes ciudades, y uno para una plataforma de simulación determinada se puede volver a utilizar siempre y cuando no haya una nueva versión de la plataforma; de la misma manera, las transformaciones pueden ser reutilizadas si sus lenguajes de modelado o textuales de origen y destino no se modifican. Dado que estos elementos incluyen toda la información involucrada en la descripción de la simulación y su refinamiento al código, facilitan la trazabilidad de los artefactos para su análisis y verificación.

Ya existen algunos trabajos de aplicación de la ISDM a las simulaciones de tráfico. No obstante, estos trabajos resultan incompletos para sustentar un desarrollo completo.

En primer lugar, existen varios LM o modelos de integración elaborados a partir de trabajos existentes en el dominio. Estos han tratado de proporcionar un marco conceptual extensible para construir progresivamente especificaciones de los fenómenos de tráfico. Sin embargo, estos trabajos generalmente resultan poco flexibles cuando se pretende modificar el marco conceptual proporcionado. Además, describen

habitualmente mediante texto las características, procesos [91] y componentes de software [97] que modelan. Muchos de estos trabajos también aportan plataformas de simulación ad-hoc, las cuales proporcionan una descripción de la semántica del marco conceptual. Sin embargo, estas descripciones no son adecuadas como definiciones de LM en ISDM, ya que su traducción a lenguajes formales no es evidente (lo que las hace inapropiadas para un procesamiento automático), y mezclan aspectos del dominio y computacionales.

Desde el punto de vista conceptual, la mayor parte de estos LM se centra en la descripción del entorno (por ejemplo, las vías por las que se circula y los vehículos). Dejan de lado normalmente los aspectos más centrados en los participantes humanos en el tráfico, tales como la percepción, la toma decisiones o el tipo de comportamiento exhibido. Estos últimos aspectos se abordan normalmente con más atención en propuestas del Modelado Basado en Agentes (MBA) [2, 35] o la Ingeniería del Software Orientada a Agentes (ISOA) [96], donde sin embargo se carece de primitivas de modelado para el tráfico.

También hay trabajos que tratan sobre la aplicación de procesos de desarrollo de ISDM a las simulaciones, pero son discusiones generales. No tienen en cuenta los aspectos específicos para llevar a cabo un desarrollo de ISDM (por ejemplo, actividades o infraestructuras) [53], o en una simulación de tráfico (por ejemplo, la escala de estos estudios o cómo tratar con componentes altamente dinámicos) [23].

Finalmente cabría destacar, dada la cercanía con la propuesta presentada en esta tesis, la existencia en el campo de la ISOA de metodologías generales de ISDM. Estas pueden ser aplicadas a problemas de simulación. No obstante, sus LM no contemplan aspectos de tráfico, por lo que es necesario hacer un esfuerzo considerable para modelarlos y procesarlos con los elementos disponibles.

1.2 Motivación

El desarrollo de simulaciones de tráfico sigue actualmente propuestas tradicionales que dificultan la construcción incremental de simulaciones, su modificación y el intercambio de información entre los expertos. Aunque la ISDM puede aportar soluciones a estos problemas, no existen en la actualidad propuestas completas en esta línea para las simulaciones de tráfico.

Esta investigación busca proporcionar una infraestructura completa de ISDM mediante la cual desarrollar simulaciones de tráfico que puedan considerar diferentes teorías, contextos y necesidades de análisis. Esto es debido a que se han detectado las siguientes carencias en las propuestas relacionadas:

- *No existe un LM general para simulaciones de tráfico que permita integrar diferentes fuentes de información y aspectos*. No se conocen LM generales que permitan realizar especificaciones de modelos que representen integradamente los elementos plasmados en diferentes teorías. Lo mismo ocurre con la especificación de los entornos donde se estudia el tráfico. Se considera además necesario abordar en el LM la especificación de los aspectos centrados en las personas que participan en el tráfico. Su comportamiento y toma de decisiones suele venir dado por sus características individuales y su conocimiento actual,

así como sus formas de interactuar con el entorno. El LM introducido en esta tesis (ver Capítulo 3) tiene en cuenta estos aspectos y los organiza en tres grupos de conceptos: *Mental*, *Entorno* e *Interactivo*. Además proporciona mecanismos de descripción de jerarquías de herencia y composición para facilitar la especificación e integración de los modelos existentes.

- *No existe un método que permita transformar los modelos generados por un LM de tráfico a código fuente y que sea intuitivo para los expertos en tráfico.* Existen lenguajes (de transformaciones y de programación) que permiten definir transformaciones para generar código fuente a partir de un modelo [14]. Estos lenguajes y sus herramientas están concebidos para ser usados por programadores y no por expertos en el dominio (en este caso el tráfico rodado). Además, en el caso de usar lenguajes de transformaciones, estos no suelen formar parte del conocimiento habitual de los programadores de simulaciones. Para abordar este problema se plantea el desarrollo de herramientas visuales adaptadas a los LM de los expertos. Estas herramientas les deben permitir realizar muchas de las tareas que habitualmente delegan en los programadores. Aquí se plantea también la integración de asistentes en estas herramientas con el objetivo de guiar a los usuarios durante el proceso.
- *No existen procedimientos centrados en las simulaciones de tráfico que guíen a los equipos de trabajo en la aplicación de marcos de desarrollo dirigidos por modelos.* Los trabajos existentes se centran en proporcionar elementos de la infraestructura (por ejemplo, LM o herramientas) u ofrecen procesos genéricos. Ello incrementa la curva de aprendizaje para estos desarrollos. Este trabajo plantea ofrecer guías concretas para la aplicación de la infraestructura anterior.
- *No se reutilizan las infraestructuras genéricas disponibles para ISDM.* Muchos de los trabajos existentes desarrollan sus propias infraestructuras ad-hoc, por ejemplo para realizar transformaciones. Cuando se usan componentes existentes, solo se hace de una forma parcial, sin aprovechar el potencial de los marcos de desarrollo completos. Esto dificulta el manteniendo y crecimiento de la base de usuarios. Este trabajo plantea hacer uso de infraestructuras estándar siempre que sea posible. En particular, este marco de desarrollo se basará en los proyectos de Eclipse para ISDM [27]. Estos proporcionan funcionalidad general para la descripción de LM, la creación de sus editores, y la manipulación de LM y especificaciones.

1.3 Objetivos

El objetivo principal de esta tesis es proporcionar una infraestructura de ISDM completa para desarrollar simulaciones de tráfico. Estas simulaciones deberán ser capaces de integrar diferentes teorías y estudios sobre el tráfico, con particular atención en el comportamiento de los individuos involucrados en el mismo, y ser fácilmente portables entre diferentes plataformas de simulación. Además, deberá ser capaz de dar soporte a diferentes necesidades de análisis. Este objetivo se descompone en los siguientes:

- *Diseñar un Lenguaje de Modelado de Tráfico (LMT).* Este lenguaje estará centrado en los individuos involucrados en el tráfico, considerando las características individuales de los mismos y sus interacciones con el entorno que les rodea, así como su toma de decisiones y el comportamiento manifestado. Para ello deberá facilitar primitivas para la descripción de los individuos y de situaciones relacionadas con el tráfico al nivel de los expertos en tráfico. También facilitará la integración de nuevas teorías, por lo que su vocabulario deberá contar con mecanismos que faciliten su extensión y modificación.
- *Desarrollar herramientas de soporte para el uso del LMT en un proceso de ISDM*. Ello requiere disponer de herramientas que faciliten la modificación del LM en caso de ser necesario, la especificación de modelos conformes a él y la transformación de los mismos, al menos para generar código fuente y su documentación asociada.
- *Elaborar guías de desarrollo para la utilización de los elementos anteriores.* Uno de los principales problemas en las propuestas de ISDM es la dificultad para utilizar adecuadamente la infraestructura proporcionada. El utilizar LM específicos requiere dar pautas para construir modelos conformes a los mismos. Del mismo modo, trabajar en base a modelos y trasformaciones requiere indicaciones sobre posibles formas de organizar el trabajo, resolver las distintas tareas y utilizar las herramientas relacionadas.

1.4 Método de trabajo

Para abordar los objetivos precedentes se ha seguido el siguiente plan de trabajo:

- *Estudio del estado del arte relativo a los marcos conceptuales relacionados con el tráfico*. Se han considerado distintas fuentes para analizar qué tipo de conceptos se emplean para estudiar o modelar situaciones de tráfico. Así, se han considerado teorías y modelos sobre tráfico relativos al comportamiento de las personas y otros elementos que participan en estos escenarios. Para profundizar en aspectos del modelado de personas, también se han analizado teorías y LM de MBA e ISOA. Desde el punto de vista de las infraestructuras, se han evaluado plataformas de simulación de tráfico y sus representaciones.
- *Estudio de la ISDM y sus herramientas.* Se han considerado las diferentes propuestas y plataformas para la definición y uso de LM y transformaciones. Aquí se ha prestado particular atención a la forma en que facilitar una mayor participación de los expertos en tráfico en las labores de desarrollo de las simulaciones.
- *Desarrollo de un LM para tráfico*. Se ha desarrollado un LM específico para describir las situaciones de tráfico, el LMT. Éste se ha organizado en torno a los individuos que participan en estas situaciones por lo que considera sus características, conocimiento, forma de relacionarse con el entorno y la manera en que toman las decisiones.
- *Implementación de herramientas de desarrollo*. Se han desarrollado tres herramientas específicas con el propósito de dar soporte al trabajo con el LMT. La primera es un editor gráfico para realizar y validar especificaciones de

modelos conforme al LMT. La segunda es un generador de código. Éste proporciona una interfaz gráfica para navegar a través de las especificaciones y mostrar información sobre sus elementos. Además, proporciona un conjunto de asistentes que simplifican algunas de las tareas más complejas de la generación de código fuente. La tercera de las herramientas es una plataforma de simulación de tráfico para realizar pruebas.

- *Extensión y validación de los elementos generados con otros estudios de tráfico*. Una vez realizadas las pruebas iniciales, se comenzaron a considerar otras teorías y plataformas para probar la propuesta. Por ejemplo, tras usar la plataforma de test se pasó a usar otras más complejas como MATSim [90] o SUMO (*Simulation of Urban MObility*) [4]. Mediante estas pruebas se comprobó la viabilidad de la propuesta efectuada en esta investigación, se evaluó la mejora de los resultados obtenidos al desarrollar simulaciones en comparación con los procesos originales, y se analizó si era necesario realizar modificaciones en la infraestructura (LMT y herramientas) para considerar necesidades previamente no identificadas.
- *Síntesis de la experiencia de desarrollo en guías*. A medida que se han elaborado los casos de estudio se han extraído pautas de uso y buenas prácticas. Éstas se han sintetizado en guías de desarrollo que incluyen diagramas de actividades para mostrar los flujos de trabajo y explicaciones para las distintas tareas.

1.5 Estructura del documento

Los resultados de esta investigación y la experimentación llevada a cabo para validarlos se recogen en los siguientes capítulos de este documento. Su organización es la siguiente:

- *Capítulo 2. Estado del arte.* Este capítulo revisa el contexto de este trabajo de investigación. Para ello profundiza en las propuestas de modelado y simulación aplicables al problema del tráfico. Aborda los estudios centrados en analizar las situaciones de tráfico (por ejemplo, entorno, vehículos y señalización), prestando especial atención a los que consideran el comportamiento de los individuos. Aunque tienen un carácter más general que la simulación de tráfico, también se analizan las contribuciones procedentes de la ISOA, incluyendo algunas de las principales metodologías basadas en agentes y sus marcos conceptuales. Finalmente se realiza una breve introducción al proceso de desarrollo de software tradicional y al basado en ISDM, revisando sus conceptos y limitaciones.
- *Capítulo 3. Lenguaje de modelado.* Se encarga de presentar el LMT. Este lenguaje es parte esencial en la investigación, ya que en él se apoyan el resto de los elementos para llevar a cabo la implementación del marco de desarrollo de ISDM. El capítulo introduce los mecanismos utilizados para su especificación, la estructura de ésta y el marco conceptual modelado. Los conceptos del LM se organizan en tres grupos (los *clústeres*) que forman el lenguaje. El clúster *Mental* toma referencias de trabajos como [79], y se encarga de representar el

estado y conocimiento actual de los individuos. Por su parte, el clúster *Entorno* representa el entorno exterior a los individuos, incluyendo sus vehículos. Se basa en el modelo Conductor-Vehículo-Entorno (CVE, *Driver-Vehicle-Environment*, DVE por sus siglas en inglés) [1]. Finalmente, el clúster *Interactivo* presenta un ciclo de percepción, razonamiento y actuación para modelar la toma de decisiones de los individuos. Está inspirado modelos de agentes procedentes de metodologías de la ISOA como Tropos [9] o INGENIAS [64]. El LMT ha sido ampliamente tratado en las publicaciones relacionadas con esta tesis (ver Sección 1.6), incluyendo las publicaciones en revista [i, ii] y de congresos [c, d, e], debido a que es el elemento principal del marco de desarrollo presentado en esta tesis. El lenguaje tiene su origen también en una publicación de congreso, [a] (ver también la Sección 1.6).

- *Capítulo 4. Herramientas de desarrollo*. Aquí se introducen las tres herramientas de desarrollo de la infraestructura para ISDM de esta investigación: el editor gráfico, el generador de código y la plataforma de test de simulación de tráfico. También se discuten algunos aspectos de su desarrollo. Estas herramientas han sido tratadas principalmente en los artículos de revista [i, ii] y en el artículo de congreso [d] (ver Sección 1.6). Una aplicación inicial de las mismas se produjo para el desarrollo de los sistemas de soporte al aprendizaje descritos en [b] (ver Sección 1.6).
- *Capítulo 5. Proceso de desarrollo.* Este capítulo describe las tareas a llevar a cabo durante el proceso de desarrollo utilizando la infraestructura propuesta en esta tesis. Este proceso permite obtener código fuente especializado para una plataforma de simulación de tráfico determinada partiendo de estudios relacionados con el dominio. Para ello se extraen las teorías sobre tráfico rodado proporcionadas por estos trabajos. Estas teorías son la base para diseñar especificaciones de modelos conformes con el LMT. Después, las especificaciones son transformadas a código fuente y especializadas para la plataforma de simulación por medio del generador de código. El capítulo ilustra con ejemplos las distintas fases del proceso de desarrollo. Este ha sido introducido en las publicaciones de congreso [c, e] (ver Sección 1.6) en donde se muestran las primeras versiones del mismo.
- *Capítulo 6. Experimentación.* Presenta dos casos de estudio que usan la infraestructura de desarrollo propuesta. El primero utiliza el modelo de comportamiento de conductores *Drivability* [5] e integra una teoría de interacción de los conductores desarrollada en esta tesis y descrita en [21]. Esta especificación se transforma y se especializa para su ejecución sobre la plataforma MATSim [90]. La nueva simulación reproduce la original, y además la extiende con un nuevo modelo de toma de decisiones en sus agentes basado en los aspectos del modelo de comportamiento. El segundo caso de estudio emplea una teoría enfocada en los factores de riesgo en el comportamiento en el tráfico de los peatones jóvenes y los conductores [78]. Se reutiliza la experimentación anterior y se extiende la teoría de interacción de los conductores a los peatones. Esta extensión se lleva a cabo basándose en las directrices aportadas por [25]. El resultado se traduce a código fuente permitiendo su utilización en la plataforma de test como librería.

- *Capítulo 7. Conclusiones finales.* Este capítulo presenta las principales conclusiones obtenidas en esta investigación y discute futuras líneas de trabajo relacionadas.

1.6 Publicaciones relacionadas

A lo largo del trabajo de investigación reflejado en esta tesis se han llevado a cabo una serie de publicaciones que se detallan en las siguientes sub-secciones.

Artículos en revistas

i. Fernández-Isabel, A., y Fuentes-Fernández, R. (2015). Analysis of intelligent transportation systems using model-driven simulations. *Sensors*, 15(6), 14116-14141. doi: http://dx.doi.org/10.3390/s150614116.

ii. Fernández-Isabel, A., Fuentes-Fernández, R. An Integrative Modelling Language for Agent-Based Simulation of Traffic. IEICE Transactions on Information and Systems, E99-D(2), páginas 1-9, 2016.

Artículos en congresos

a. Fernández-Isabel, A., y Fuentes-Fernández, R. (2011). An agent-based platform for traffic simulation. En E. Corchado, V. Snášel, J. Sedano, A. E. Hassanien, J. L. Calvo, D. Ślęzak (Eds.), *6th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2011)*, Advances in Intelligent and Soft Computing 87, 505-514. Springer-Verlag.

b. Blanco, D., Fernández-Isabel, A., Fuentes-Fernández, R., y Guijarro, M. (2012). Using multi-agent systems to facilitate the acquisition of workgroup competencies. En J. Bajo Pérez, J. M. Corchado Rodríguez, E. Adam, A. Ortega, M. N. Moreno, E. Navarro, B. Hirsch, H. Lopes-Cardoso, V. Julián, M. A. Sánchez, P. Mathieu (Eds.) *10th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2012)*, Advances in Intelligent and Soft Computing 156, 223-230. Springer-Verlag.

c. Fernández-Isabel, A., y Fuentes-Fernández, R. (2015). A model-driven engineering process for agent-based traffic simulations. En M. S. Obaidat, J. Kacprzyk, T. Ören (Eds.), *5th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2015)*, 418-427. SCITEPRESS.

d. Fernández-Isabel, A., y Fuentes-Fernández, R. (2015). Developing an integrative modelling language for enhancing road traffic simulations. En M. Ganzha, L.

Maciaszek, M. Paprzycki (Eds.), *9th International Workshop on Multi-agent Systems and Simulation (MAS&S 2015)* en *2015 Federated Conference on Computer Science and Information Systems (FedCSIS 2015)*, 1759-1770. IEEE.

e. Fernández-Isabel, A., y Fuentes-Fernández, R. (2015). An integrative framework for the model-driven development of traffic simulations. En *16th Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 2015)*, 1-10.

Capítulo 2. Estado del arte

Este capítulo introduce los distintos aspectos considerados en el modelado y análisis de las situaciones de tráfico. Dado el foco de este trabajo en las personas que participan en el fenómeno, se centra en el estudio del comportamiento humano en el tráfico rodado. Se consideran trabajos que cubren desde la inferencia del comportamiento o la interpretación de gestos de individuos, a la observación del comportamiento social tanto de conductores como en los grupos de peatones. También se abordan los distintos modelos de comportamiento que se han usado para implementar estos aspectos en simulaciones de tráfico. Se revisa en particular el uso de agentes inteligentes para representar a los individuos implicados y teorías sociales para representar grupos. Después se analiza cómo estos conceptos se han plasmado en algunas de las plataformas de simulación más conocidas. Finalmente, este capítulo considera algunas metodologías de desarrollo presentes en la literatura asociada, centrándose en especial en la ISDM.

2.1 Introducción

El marco de desarrollo de ISDM propuesto en esta tesis tiene dos elementos fundamentales, el LMT y el proceso de desarrollo, cuyo uso se soporta con herramientas. Su correcto desarrollo requiere conocer los requisitos de los expertos, los conceptos que manejan en su trabajo, y la forma en que proceden en el desarrollo de sus simulaciones. Este capítulo de la tesis analiza cómo se han tratado estos aspectos en la literatura y herramientas.

La Sección 2.2 presenta los marcos conceptuales. En muchos casos estos trabajan a diferentes niveles de granularidad, desde los más finos (a nivel de los componentes de individuos o vehículos) hasta los más gruesos (que consideran grupos de elementos o propiedades generales de los flujos de tráfico). Por su parte, la Sección 2.3 presenta modelos de desarrollo de simulaciones. Estos son guías a seguir en los proyectos junto con una serie de artefactos de soporte, como pueden ser LM, herramientas software, o métodos de análisis. En este caso se distinguen modelos tradicionales organizados alrededor de la elaboración de código, y los de ISDM centrados en la elaboración de modelos. En esta última sección también se tratan las infraestructuras más populares usadas para la ISDM.

2.2 Marcos conceptuales

Esta sección presenta distintos trabajos usados para modelar y analizar fenómenos del tráfico. Aparecen múltiples modelos con niveles de granularidad diferentes, desde los centrados en los individuos a los centrados en grupos de los mismos. En función de estos niveles, los modelos se clasificarán en microscópicos (centrados en los participantes individuales en el tráfico), macroscópicos (centrados en grupos de participantes o flujos de tráfico) y mesoscópicos (que combinan características de los dos tipos anteriores).

La organización de esta sección es la siguiente. La Sección 2.2.1 profundiza en los estudios realizados sobre personas reales para establecer factores que influyen en su comportamiento en el tráfico. Después, la Sección 2.2.2 se encarga de introducir los modelos teóricos más importantes para la representación de las personas en el tráfico para su análisis y simulación. En la Sección 2.2.3 se discuten los distintos tipos de simulaciones conforme a los diferentes niveles de granularidad. Finalmente, la Sección 2.2.4 aborda la aplicación de las metodologías de la ISOA para la simulación de los individuos y su comportamiento.

2.2.1 Estudios reales de comportamiento

Estos trabajos son realizados sobre individuos en entornos reales (por ejemplo, dentro de un vehículo) para obtener información directa del comportamiento de los mismos. De esta manera se pueden obtener clasificaciones de tipos de personas e identificar sus patrones de comportamiento. Estos patrones son útiles por ejemplo para predecir el próximo movimiento del individuo (como la realización de un cambio de carril sin indicarlo previamente) o para poder señalar a las personas más propensas a sufrir algún tipo de percance en el tráfico.

La Sección 2.2.1.1 introduce la observación e interpretación de las señales físicas de los individuos. Estas interpretaciones permiten realizar las clasificaciones de los mismos. Por su parte, la Sección 2.2.1.2 aborda la inferencia del comportamiento de las personas involucradas en el tráfico, identificando los patrones que denotan una futura acción de la persona sin que ésta la indique explícitamente.

2.2.1.1 Observación e interpretación del comportamiento

Para la interpretación de los gestos y las decisiones de los individuos se utilizan múltiples instrumentos que recogen la información de las actuaciones y las señales físicas. Existen diferentes posibilidades según el tipo de individuo observado y los instrumentos que se utilicen.

En los conductores, una primera opción consiste en medir la presión ejercida sobre los pedales del freno y el acelerador del vehículo, consiguiendo reconocer el tipo de maniobras realizadas por cada conductor. Hay varios ejemplos de este tipo de aproximaciones, como [46, 94].

En [94] se desarrolla un Controlador de Articulación de Modelo Cerebelar (CMAC, *Cerebellar Model Articulation Controller*) con el objetivo de considerar los datos procedentes de los sensores colocados en los pedales de los vehículos. Se trata de obtener información específica sobre el estrés o agresividad del individuo que está al

volante. Clasificando y evaluando estos datos se elabora un perfil de comportamiento que permite determinar el estado de ánimo del conductor: los individuos que presionan con más fuerza y frecuencia los pedales se identifican como más agresivos, mientras que los que realizan presiones menos enérgicas y repetitivas sobre estos se les reconoce como más moderados. Recientemente se está desarrollando un método para no sólo identificar de manera precisa la emoción del conductor, sino también poder predecir con cierta exactitud sus emociones. Esta información emocional sobre los conductores resulta relevante en el modelado y simulación de su comportamiento, ya que la toma de decisiones durante la conducción se ve afectada.

Por su parte, [46] añade a la información sobre el uso de los frenos la gesticulación y los movimientos que realiza el conductor durante la conducción. La información obtenida al recoger y depurar estos datos permite a los autores inferir el tipo de comportamiento del conductor e identificar situaciones potencialmente peligrosas. Los resultados obtenidos en esta aproximación muestran una relación directa entre el comportamiento de los conductores y los factores externos que deben afrontar, como pueden ser congestiones y tráfico lento, vías con poca visibilidad o la influencia de otros conductores próximos.

Existen otras opciones de información de entrada en este tipo de estudios. Por ejemplo, en [36] el análisis se basa en el estudio del diálogo del conductor. A partir de él se trata de establecer las emociones que experimenta, ya que estas afectan a su modo de conducir. Este sistema se apoya en una serie de funciones matemáticas lineales. Además, utiliza una base de datos que contiene un conjunto de datos relacionados con experimentos que analizan el comportamiento de los conductores basándose en el estado de ánimo de los mismos. Estos datos junto a las funciones diseñadas sirven para desarrollar reglas de entrenamiento para una red neuronal borrosa, la cual identifica los distintos tipos de emociones en los conductores.

El comportamiento de las personas también puede verse afectado por los otros individuos que les acompañan en el trayecto. Por ello hay trabajos que consideran las relaciones de los conductores con los acompañantes o la influencia de otros peatones sobre un individuo desplazándose a pie.

En [17] se tiene en cuenta la presencia de los pasajeros junto a otras características en el aumento del riesgo de colisión entre vehículos. Se evalúa la facilidad de los pasajeros para distraer a los conductores e incluso modificar algunas de sus características individuales como pueden ser el carácter o el nivel de estrés. Estas modificaciones son causadas por el rol que toman los pasajeros, pudiendo ser beneficioso o perjudicial para el conductor.

En [19] se estudia la influencia de los grupos de peatones sobre los individuos. Se consideran los datos obtenidos de múltiples observaciones realizadas sobre un entorno que presenta riesgos para los peatones. Se trata de un cruce donde interactúan peatones y conductores. Se elabora una simulación basada en datos reales con la intención de obtener estadísticas y conclusiones relevantes basadas en la probabilidad de las acciones de los individuos. Además, se introduce el concepto *grupo de peaton*es. Éste se basa en la siguiente suposición: un peatón que está al frente de un grupo tiene influencia sobre el resto de individuos que le siguen. Por ejemplo, si este primer individuo comienza a atravesar una vía, el resto de individuos comenzarán a hacerlo también suponiendo que el peligro representado en este caso por los vehículos es suficientemente bajo como para poder cruzar con seguridad. Esto muestra como el

comportamiento de los peatones se ve modificado por las personas a su alrededor, dejándose llevar por las decisiones de otros en ciertas ocasiones.

Las conclusiones obtenidas en este grupo de estudios señalan algunos de los aspectos a considerar en el LMT y en el modelado de escenarios concretos. Estos trabajos indican que el comportamiento de las personas involucradas en el tráfico no se ve afectado sólo por los elementos externos, sino que también depende de su estado de ánimo y su actitud al enfrentarse a estos elementos y situaciones. Además, se muestra la influencia de la componente social.

2.2.1.2 Inferencia del comportamiento

Este tipo de estudios intentan inferir el comportamiento de las personas para predecir con un cierto nivel de certidumbre algunas de las maniobras más simples que llevan a cabo. Las maniobras concretas y cómo se realiza dicha predicción cambian en función de cómo está enfocado cada uno de los trabajos.

Así, en [87] se presenta un sistema capaz de anticipar con una probabilidad elevada cuando un conductor va a realizar un cambio de carril. Para ello, aplica un modelo Gaussiano [70] de tipo estático condicional a una red Bayesiana [55]. En cuanto a las maniobras de cambio de carril, estas se clasifican en tres tipos según la influencia de los factores externos en los conductores:

- Permanencia en el carril actual.
- Cambio normal al carril más próximo.
- Cambio de emergencia al carril más próximo.

Además, este trabajo define perfiles de comportamiento para los conductores mediante una clasificación jerárquica en cuatro niveles:

- Nivel 1: Dominio de las maniobras del vehículo. Función ejecutiva.
- Nivel 2: Manejo de las situaciones del tráfico. Función específica de la situación.
- Nivel 3: Objetivos y contexto de la conducción. Función específica del dominio del tráfico.
- Nivel 4: Objetivos durante la vida y habilidades desarrolladas. Función independiente del dominio del tráfico.

Este tipo de trabajo permite comprender mejor las interacciones entre los individuos involucrados en el tráfico. En concreto, es útil para saber cómo los conductores interpretan las señales externas para anticipar cambios en el entorno y planificar así sus propias acciones.

Otro trabajo similar que analiza e infiere el comportamiento es [77]. En este caso se emplean sensores para medir la presión sobre los pedales de manera similar a los trabajos de la sección anterior (ver Sección 2.2.1.1) y algoritmos probabilísticos mediante modelos ocultos de Markov [67]. Al igual que el caso anterior, se anticipan los cambios de carril, aunque también se consiguen inferir los giros, la posición del vehículo y el estado de ánimo del conductor mediante las señales biométricas obtenidas por los sensores.

Por otra parte, el trabajo también muestra una clasificación precisa del estado del conductor (por ejemplo, neutral, agresivo, distraído y adormilado). Esta clasificación proporciona un punto inicial a considerar en aproximaciones que modelen el

comportamiento de los conductores, ya que permite la obtención de perfiles temporales de comportamiento.

Una aproximación distinta a las dos anteriores es la de [42]. En este caso se infiere el comportamiento de los conductores al llegar a un cruce. La dificultad radica en la influencia mutua entre los vehículos: se considera que los conductores tienen otros vehículos delante de ellos, por lo que la velocidad de todos ellos es similar y se ve afectada por los cambios en la de los demás. Para llevar a cabo esta inferencia se utilizan redes Bayesianas [55] basadas en un Modelo Inteligente del Conductor (IDM, *Intelligent Driver Model*), mientras que los datos a evaluar son obtenidos mediante el Sistema de Posicionamiento Global (GPS, *Global Positioning System*). En cuanto a los tipos de acciones evaluadas por este trabajo son:

- Continuar de frente.
- Parar en línea de stop.
- Girar a la derecha.
- Girar a la derecha y parar en un paso de peatones.

Las maniobras y perfiles utilizados en este tipo de trabajos son importantes para la realización del LMT, ya que indican información a considerar por el mismo. Además, muestran que el comportamiento de las personas es predecible en base a ciertos indicadores, y que entre estos indicadores se ha de considerar la influencia del resto de individuos.

2.2.2 Modelos teóricos de comportamiento

Para representar el comportamiento de las personas involucradas en el tráfico en simulaciones y otros medios de análisis se han desarrollado una gran cantidad de modelos. Estos pueden clasificarse en diferentes grupos según los elementos considerados por cada uno de ellos. Aquí se establece una clasificación general que diferencia entre los modelos taxonómicos, que establecen una clasificación de determinados grupos de atributos y sus valores (ver Sección 2.2.2.1) y los modelos funcionales, que se centran en cómo se modela la toma decisiones y acción (ver Sección 2.2.2.2).

2.2.2.1 Modelos taxonómicos

Las aproximaciones que utilizan este tipo de modelos suelen producir clasificaciones descriptivas de ciertos elementos del tráfico en base a un contexto y el conjunto de elementos relacionados con el mismo. Estos modelos se clasifican en dos tipos: modelos de características y modelos de análisis de tareas.

En un modelo de características se establece una correspondencia y clasificación entre ciertas características observables de personas y su entorno y otras que al no ser observables directamente se derivan de las anteriores. Por ejemplo, se representan mediante variables los rasgos personales de los individuos como el nivel de estrés, la edad o el género, o del entorno como la velocidad o la presencia de pasajeros. Entre las características que se derivan están por ejemplo el nivel de agresividad de la conducción o el riesgo de accidente.

Existen múltiples ejemplos de modelos de características en la literatura, como [11, 84]. En el caso de [11] se desarrolla un modelo relativo a la motivación y la

personalidad de los individuos para evaluar la conducción agresiva y las distracciones en el tráfico de los conductores. Las conclusiones obtenidas indican que ambas situaciones producen efectos similares sobre el comportamiento. En [84] se introducen taxonomías para definir los posibles errores que se pueden cometer durante la conducción. Estos errores se clasifican según unos conjuntos de factores identificados como posibles causantes de accidentes. A su vez, estos factores se organizan según su tipología y origen (por ejemplo, factores del vehículo o las condiciones físicas del individuo).

Por otro lado, los modelos de análisis de tareas describen las habilidades o tareas requeridas (tanto aprendidas como innatas) para llevar a cabo actividades en el tráfico. No se trata de modelos de actuación (que describan el ciclo de acciones), sino de listados o taxonomías de tareas involucradas en la conducción o el movimiento.

Entre los modelos de análisis de tareas están [20, 80]. En [80] se realizan comparaciones entre las tareas desempeñadas por el conductor en el cambio de marchas manual y automático. Los resultados muestran que la ejecución de esta tarea depende de la experiencia de los individuos en la conducción. Ésta a su vez está relacionada con la necesidad de disponer de conocimientos de aplicación automatizada para llevar a cabo la tarea con la velocidad adecuada. El trabajo en [20] clasifica múltiples tareas relacionadas con el tráfico. Estas se dividen en tres grupos principales: tareas de dirección longitudinal, tareas en intersecciones y otro tipo de tareas. Estas últimas a su vez se basan en otras tareas que presentan ciertos requisitos de comportamiento como percepción y memoria del individuo, o el manejo del vehículo. Además se describen y evalúan las diversas situaciones que pueden ocurrir durante la ejecución de estas tareas teniendo en cuenta los comportamientos conscientes y no conscientes. Las conclusiones obtenidas muestran que este modelo permite introducir prácticamente cualquier otro tipo de tarea relacionada con la conducción sin apenas modificaciones.

Respecto a la influencia de este tipo de modelos en el LMT, son la base para establecer las acciones que pueden llevar a cabo las personas en las simulaciones. Debido a la variedad de trabajos, y su constante evolución, el objetivo no puede ser proporcionar un conjunto exhaustivo de primitivas para modelar todas las características y acciones. Por el contrario, se ha de proporcionar un conjunto limitado de primitivas flexibles que permitan extender el lenguaje con los distintos elementos en estas categorías.

2.2.2.2 Modelos funcionales

Estos modelos se clasifican en tres tipos: modelos mecánicos, modelos de control adaptivo y modelos cognitivos.

Los modelos mecánicos se basan en la representación de las influencias mutuas entre elementos inspiradas en principios mecánicos (por ejemplo, seguimiento del predecesor) o físicos (generalmente como fuerzas de atracción o repulsión). Por tanto, no se centran tanto en el modelado del comportamiento como en el flujo de elementos. Esto implica que las características individuales de los conductores se desdibujan en el conjunto.

Existen múltiples ejemplos de estos modelos [26, 30, 48, 49]. En [26] se desarrolla un modelo de un único flujo continuo de vehículos. Éste aplica principios basados en la dinámica de fluidos para obtener relaciones entre la velocidad y progresión del conjunto, la densidad del flujo y la fluidez del mismo. En [49], la densidad del tráfico

y la velocidad del conjunto de vehículos se modelan aplicando análisis estadísticos y modelos matemáticos. Los modelos mecánicos también se aplican a flujos de peatones. Por ejemplo, en [48] se aplican ecuaciones de movimiento en el campo magnético considerando como polos positivos cada peatón u obstáculo, y polos negativos el destino deseado por cada individuo. De este modo, los peatones se mueven en dirección a su destino y evitan posibles colisiones. Para calcular su velocidad se tiene en cuenta la suma de las fuerzas producidas por los destinos, obstáculos y otros peatones. En [30] se emplean principios similares a los de [48], pero en este caso las fuerzas consideradas son las llamadas sociales, que están relacionadas con la motivación del individuo para alcanzar su destino.

Este tipo de modelos se aplican fundamentalmente en modelos macroscópicos, ya que se centran en los aspectos observables externamente a nivel de grupos. En los otros casos (modelos microscópicos y mesoscópicos), podrían considerarse desde una perspectiva de influencia. Esto es, en el sentido de que el conjunto de individuos condiciona algunas características de un individuo.

Los modelos de control adaptativo recogen información del entorno externo (por ejemplo, señales de tráfico o presencia de un vehículo), en función de la cual producen las salidas que determinan el comportamiento del elemento (por ejemplo, vehículo o peatón). Se pueden enfocar en el control de los vehículos o en el flujo del movimiento. En el primer caso, estos modelos se centran en la correspondencia entre las señales de entrada que representan los cambios de la vía o el entorno (por ejemplo, curvas o límites de velocidad) y las acciones de conducción, mientras que en el segundo caso se centran en modelar el flujo de decisiones y acciones que permiten procesar la información.

Ejemplos de modelos enfocados al control de vehículos son [44, 75]. En [44] se describe un sistema de control lineal de lazo cerrado para simular la conducción de un vehículo mediante el volante. Para ello se desarrollan fórmulas que tienen en cuenta las múltiples variables necesarias en este proceso, y las diversas situaciones que pueden ocurrir durante la conducción. Otra aproximación de control de vehículos con un modelo similar es [75]. En ella se utiliza un modelo inspirado en conductores reales con dos puntos de referencia para orientarse en la vía. Además, permite realizar maniobras correctivas en caso de necesidad, cambios de carril sin modificar el comportamiento del vehículo, y capturar algunos aspectos individuales de los conductores (en particular la ejecución de maniobras).

Este tipo de modelos del comportamiento se suelen restringir a detallar maniobras en estudios sobre individuos. Por motivos computacionales, resulta prácticamente inviable considerar simulaciones con numerosos participantes donde sus acciones se consideran a este nivel de detalle. Cuando el foco del estudio está en los aspectos *sociales* del tráfico (la influencia mutua entre sus participantes), se suelen considerar al menos maniobras completas y no acciones más básicas.

Respecto a los modelos de control del flujo del movimiento, un ejemplo es el trabajo descrito en [39]. En él se introduce un flujo de conducción (ver Figura 1) que permite modelar el proceso llevado a cabo para atravesar una intersección sin señalización alguna. Este flujo permite preguntar por una serie de circunstancias y según el tipo de respuesta obtenido se decide tomar una de las posibilidades ofrecidas con el fin de conseguir un comportamiento apropiado.

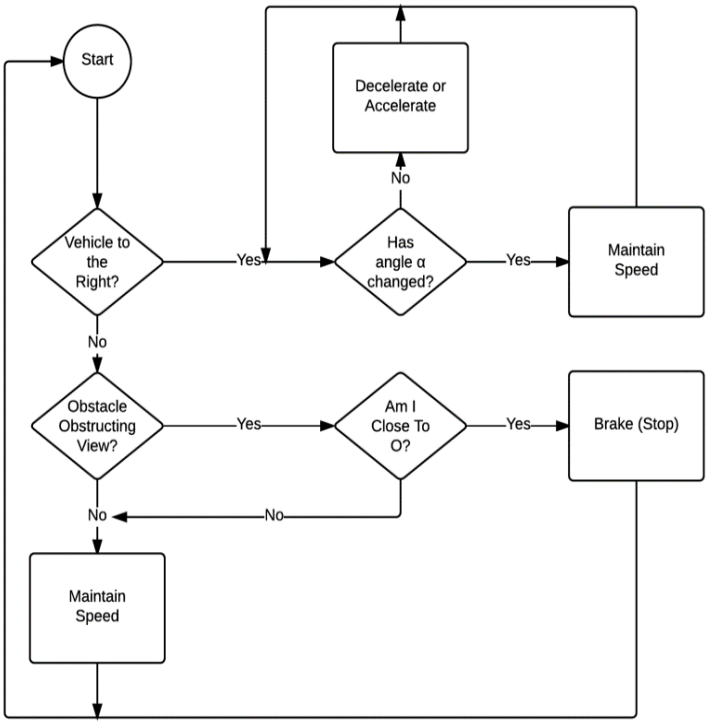

Figura 1: Modelo de flujo de aproximación a una intersección [39].

Este tipo de modelos tienen influencia sobre el proceso de toma de decisiones de las personas en este trabajo de tesis. Se trata de modelos que pueden representarse mediante la evaluación de objetivos vinculados a las tareas.

Con vistas a simplificar la complejidad computacional de estos modelos, es frecuente adoptar un modelo más abstracto que sólo considera aspectos de movimiento de un elemento (por ejemplo, vehículo o peatón) y obvia maniobras concretas o tomas de decisiones complejas. Aquí resultan muy populares diversos modelos de autómatas. Estos representan a los elementos activos del tráfico como autómatas (con reglas simples para cambiar ciertos atributos) situados en celdas del entorno. Ejemplos de ellos son [18, 25].

El modelo de coste-beneficio en [25] representa a los individuos (en general peatones) como partículas en celdas. Cada celda es adyacente a otras nueve y puede ser ocupada a lo sumo por un individuo. A cada celda se le asigna una puntuación. Esta puntuación representa la ganancia conseguida por cada individuo cuando se desplaza hacia su destino. Por otro lado, se añade un efecto de repulsión entre individuos para evitar su superposición. El movimiento de los individuos se realiza hacia la celda cuyo valor le proporciona el máximo beneficio.

En [18], los autómatas modelan el movimiento a través de las celdas. La ocupación de una celda depende de reglas locales, las cuales se actualizan cada cierto tiempo. Cada movimiento de los individuos incluye tanto el cambio de carril como de celda. En cada intervalo de tiempo, cada celda puede tomar uno de dos estados: ocupada y desocupada.

También en esta línea de autómatas pero considerando relaciones más complejas aparecen los modelos de encolado de redes como [88]. Este tipo de modelos de comportamiento se emplea habitualmente para simular la evacuación de grupos de individuos, por ejemplo, en incendios de edificios. El funcionamiento consiste en la aplicación del método de Monte Carlo [74] discreto, donde cada habitación es un nodo y las puertas entre habitaciones son enlaces. Cada individuo parte de un nodo, es encolado en un enlace, y al atravesarlo llega a otro de los nodos. Cada peatón tiene un destino deseado, de manera que su objetivo es moverse de su posición actual al destino lo más rápido posible.

Por su parte, los modelos cognitivos se pueden ver como un paso más allá de los modelos de control de flujo del movimiento, donde se hace explícita la motivación de los participantes y su forma de pensar. En este tipo se engloban buena parte de los modelos de agentes, por ejemplo los basados en el modelo de Creencias-Deseos-Intenciones (CDI, *Beliefs-Desires-Intentions*, BDI por sus siglas en inglés) [69]. Ejemplos de estos modelos son [16, 54].

En [54] se describe un modelo (ver Figura 2) basado en la teoría de la autodeterminación de los individuos [15]. Esta teoría supone que cada individuo presenta sus propias orientaciones motivacionales generales. Se distinguen dos tipos de orientaciones: por estrés o presión, y ego-defensiva. La primera está relacionada con las características del entorno mientras que la segunda se centra en la defensa de los propios sentimientos o intereses del individuo. Sus conclusiones indican que hay ciertas propiedades de los individuos al volante (por ejemplo, nivel de estrés o enfado) que cuando varían modifican el comportamiento de los mismos incrementando el riesgo de accidente.

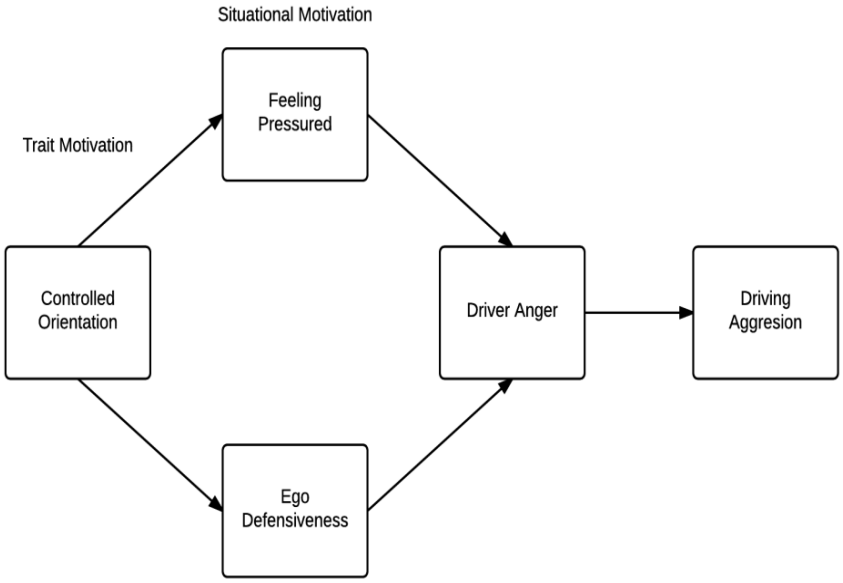

Figura 2: Modelo motivacional de conducción agresiva [54].

Por su parte, en [16] se consideran agentes cognitivos que tienen un estado mental con varios elementos: creencias, capacidades, compromisos, y normas de comportamiento y compromiso. Sobre este estado implementa un modelo basado en los trabajos clásicos de [81]. La toma decisiones de los conductores (ver Figura 3) la basa en la teoría desarrollada en [7]. Ésta afirma que los conductores tienen una serie de objetivos que cumplir entre un origen y un destino dentro de un período de tiempo determinado, mientras intentan que el coste de ese desplazamiento sea el más bajo posible en términos de distancia o tiempo de viaje. Para ello adquieren información sobre el entorno a través de su observación directa o indirecta (por ejemplo, navegadores). Una vez obtenida esa información, se procesa e interpreta conforme a su conocimiento actual. A su vez, los conductores presentan sus propias preferencias (por ejemplo, rutas y horarios de salida) debido a su percepción, restricciones y características individuales.

Este tipo de teorías ofrecen un modelo completo de la acción de las personas en el tráfico. Permiten integrar en un ciclo la percepción (adquisición de información del entorno), reflexión (generación de nueva información a partir de la disponible) y actuación (ejecución de acciones sobre el entorno). Ello permite contemplar la mayor parte de las aproximaciones en el área como especializaciones de estas. Además apuntan a la necesidad de incluir relaciones de influencia con la motivación en el modelado de los participantes en el tráfico. La razón es que estos objetivos de alto nivel afectan a la actuación inmediata de las personas.

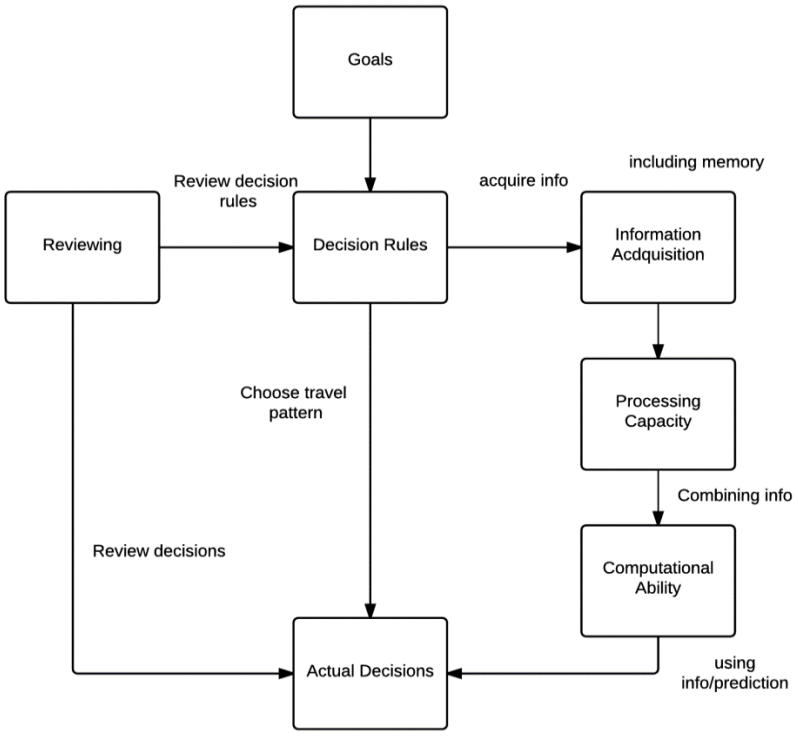

Figura 3: Proceso original de decisión de los conductores [16].

La revisión anterior se ha centrado en el modelo general de actuación. Otro aspecto importante sobre el que evaluar estas propuestas es la información concreta que consideran. Aquí se puede encontrar también una amplia variedad de modelos que incluye, entre otros, los modelos de compensación de riesgos, de teoría de umbral de riesgo y de evitación de amenazas.

Los modelos de compensación de riesgo como [95] consideran que la toma inmediata de decisiones se basa en el riesgo percibido. En ellos, cuanto mayor es el riesgo percibido, menor es la velocidad elegida por el conductor.

Los modelos de teoría del umbral de riesgo introducen un valor *umbral* a partir del cual el comportamiento del individuo puede verse modificado. Estas modificaciones pueden ser similares a las ya mencionadas para los modelos de compensación. Un ejemplo de estos modelos es [52]. En él, el umbral se calcula a partir de la relación entre el riesgo percibido en el tráfico y el nivel de probabilidad subjetiva de un evento peligroso junto a la importancia subjetiva de las consecuencias del mismo. Se señala que esta percepción de riesgo puede ser modificada mediante ciertos elementos, tales como la educación del individuo o campañas de concienciación.

Por último, los modelos de evitación de amenazas se centran en cómo se aprende a evitar los riesgos. Consideran que la experiencia de un individuo que ha sufrido algún tipo de riesgo subjetivo es aversiva, y por tanto los conductores intentarán escapar de las situaciones que les recuerden ese tipo de experiencia. Un ejemplo de estos trabajos es [24]. En él, los conductores realizan una conducción defensiva tendente a reducir los peligros. Este esquema permite modelar las reacciones anticipatorias de los conductores.

2.2.3 Tipos de simulaciones

Las simulaciones de tráfico y las plataformas empleadas en su desarrollo contemplan desde modelos microscópicos (centrados en representar a los individuos involucrados en el tráfico) a macroscópicos (centrados en los flujos que producen los individuos moviéndose en grupos). Los tipos de individuos considerados suelen ser conductores y peatones. Los acompañantes y las relaciones de influencia de estos sobre los conductores no suelen ser tenidas en cuenta.

Las siguientes secciones analizan ejemplos de plataformas y simulaciones siguiendo las dimensiones anteriores. La Sección 2.2.3.1 introduce las simulaciones microscópicas y la Sección 2.2.3.2 las macroscópicas de conductores y peatones. La sección cierra con las simulaciones híbridas o mesoscópicas en la Sección 2.2.3.3.

2.2.3.1 Simulaciones microscópicas

Las simulaciones de tráfico microscópicas son sistemas que presentan un modelo de tráfico enfocado en los individuos. Estos individuos pueden ser representados mediante algunos de los modelos vistos anteriormente, como los autómatas celulares o los agentes (ver Sección 2.2.2). Ejemplos de este tipo de simulación son [33, 62].

El trabajo en [62] presenta pequeños grupos de conductores que interactúan en un cruce o intersección donde no hay ninguna señal de tráfico. Esto facilita que los agentes que simulan estos individuos interactúen entre ellos de manera libre, incrementando su velocidad y obligando al resto a frenar, o frenando y cediendo el paso. Estas decisiones se hacen depender del estado mental de cada agente.

Este trabajo también lleva a cabo una clasificación simple de ciertas características individuales de los conductores mediante la introducción del *principio egoísta*. Esta clasificación se fundamenta en la suposición de que todo individuo tiene un cierto grado de egoísmo por el cual desea cumplir su objetivo principal. En este caso, este objetivo consiste en llegar lo más rápido posible al destino deseado de la forma más segura posible. Además, se introduce un atributo que representa la impaciencia del conductor. Este atributo puede modificarse de manera dinámica durante la simulación, pudiendo tomar tres valores distintos: cauteloso, normal y agresivo. La conducta de los individuos se completa con la introducción de un atributo que simboliza el factor de ímpetu de los mismos cuando realizan adelantamientos. Éste se encarga de valorar el nivel de seguridad de cada conductor en sí mismo, influyendo en las decisiones a tomar para completar un adelantamiento o volver al carril en el que se encontraba para intentarlo cuando las condiciones hayan mejorado.

En cuanto a los objetivos que los agentes deben satisfacer, se consideran dos tipos: macro y micro objetivos. Los primeros evalúan el destino que los agentes desean alcanzar y la ruta seleccionada para ello, mientras que los segundos se centran en la toma de decisiones en cada paso intentando satisfacer los primeros.

Este trabajo tiene una gran influencia en el desarrollo de la teoría de interacción de los individuos empleada en esta tesis. La estructura de objetivos propuesta sigue la organización anteriormente descrita fundamentada en macro y micro objetivos. Además, el LMT incorpora primitivas para modelar las características individuales (por ejemplo en [62] la impaciencia del conductor y el factor de ímpetu).

En [33] se realiza una simulación sobre un entorno real. Se persigue estudiar el efecto de colocar unos tornos de entrada/salida en estaciones de tren del área metropolitana de Lisboa. El objetivo de los tornos es regular el flujo de los individuos a pie. Además, se debe evitar que las personas puedan acceder a los trenes sin haber pagado el correspondiente billete, y permitir una evacuación de la estación en caso de que se produzca alguna situación de riesgo. Para llevar a cabo esta aproximación se emplea el modelo de simulación microscópica NOMAD [32] (ver Figura 4), generando diferentes rutas de individuos y comportamientos variados. NOMAD está basado en actividades, lo que implica que las acciones de los peatones están determinadas en gran medida por las actividades que éstos han planeado realizar. Entre estas actividades se encuentra la elección de la ruta y la elección del área de actividad, las cuales dependen de las condiciones de tráfico. Esto significa que cuando alguna ruta se congestiona, los peatones pueden evitarla siempre que haya alguna alternativa aceptable disponible. Además, el modelado de las tareas operacionales durante el desplazamiento del individuo (por ejemplo, la aceleración o los procesos de interacción) se basa en datos reales de las estaciones y en una teoría específica sobre el comportamiento de peatones [31].

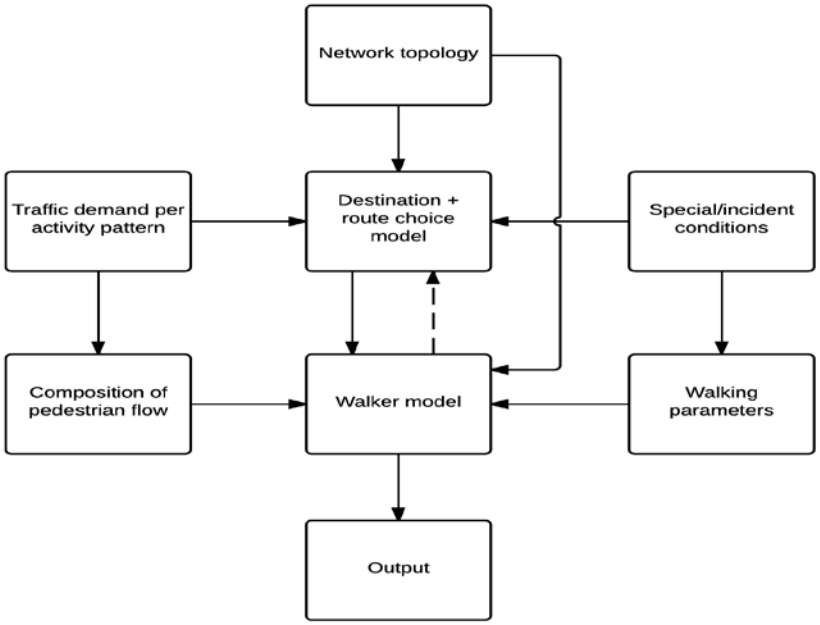

Figura 4: Esquema original del modelo NOMAD [32].

Algunas plataformas de simulación populares como MATSim [90] y SUMO [4], también pueden considerarse en este grupo. Ambas están orientadas a agentes y emplean una representación en forma de grafo del entorno. Permiten especificar participantes individuales en la simulación y también características de grupos de participantes y flujos de los mismos. Estos últimos no obstante se representan en términos de agentes individuales.

Como se desprende de los ejemplos, este tipo de simulaciones son implementaciones bastante directas de los modelos teóricos revisados en la Sección 2.2.2. Son por tanto modelos que al igual que los anteriores pueden aportar conceptos adicionales y nuevos modelos de toma de decisiones y acción.

2.2.3.2 Simulaciones macroscópicas

Este tipo de simulaciones representan el comportamiento de los individuos en su conjunto, sin considerar las características individuales de los mismos. Por este motivo, suelen utilizarse en la representación de grandes áreas de terreno o zonas muy saturadas de personas. Existen múltiples ejemplos de implementaciones de este tipo de modelos para tráfico, como [3, 92].

En [92] se desarrolla un modelo macroscópico para el tráfico urbano y las carreteras. Para el primer entorno utiliza un modelo basado en [37], en donde la trayectoria de cada vehículo se calcula de manera individual. Este modelo puede emplearse con tráfico fluido o en retenciones, permite describir efectos de bloqueo entre flujos de individuos, y presenta colas horizontales para gestionar las congestiones en las vías urbanas. Para el segundo se basa en el modelo METANET [50]. Éste representa las carreteras mediante un grafo donde cada nodo dispone de las mismas características y cada arista

es dividida en segmentos de un tamaño determinado. El estado de cada segmento es caracterizado por la densidad del tráfico (es decir, el número de vehículos por carril y unidad utilizada para medir el tamaño de los segmentos), la velocidad media y la cantidad de vehículos que abandonan el segmento por unidad de tiempo. Este modelo es mejorado con una serie de consideraciones nuevas, como los límites de velocidad o la descripción de los diferentes efectos de una densidad descendente positiva o negativa del gradiente sobre la velocidad. Las fórmulas de ambos modelos se combinan dependiendo del entorno donde se lleve a cabo la simulación. Esto hace que el punto fuerte de este trabajo sea su capacidad de adaptación y flexibilidad para simular los flujos de tráfico de cualquier tipo de entorno.

Un ejemplo de simulación macroscópica basada en peatones se introduce en [3]. En esta aproximación se analiza el flujo producido por los individuos a pie en entornos muy masificados, como puede ser el trayecto desde una parada del transporte público hasta el lugar donde se produce un evento de música o futbol. Para ello se emplea un modelo macroscópico que describe el movimiento de los peatones basado en [66]. Este modelo asume que los individuos se desplazan en el sentido de acumulación generada procurando llegar al lugar donde se encuentra el evento, sin considerar otros posibles caminos alternativos e intentando evitar los lugares más masificados. Para calcular la densidad de individuos en determinados lugares y en un momento concreto utiliza ecuaciones dinámicas. Esta densidad puede ser limitada a un número de personas por metro cuadrado y se asume una distribución uniforme de los individuos. Respecto a la representación del entorno, se lleva a cabo mediante grafos que presentan una dirección determinada. Las zonas más complejas donde pueden producirse embotellamientos, tales como puertas y alrededores, son tratadas con mayor detalle para mejorar el nivel de realismo. Para probar este modelo macroscópico, se integra su simulación junto a la de un sistema de control de acceso a una estación de metro.

2.2.3.3 Simulaciones mesoscópicas

Las simulaciones mesoscópicas son sistemas híbridos. Se centran en los individuos y/o en los grupos que forman, estudiando sus interacciones y características a distintos niveles de detalle.

Un ejemplo típico de simulación mesoscópica es [89]. En ella se introduce un modelo que considera dos tipos de grupos de peatones: grupos físicos (también llamados grupos reales), los cuales corresponden al número de personas en determinados lugares, y grupos virtuales o lógicos, encargados de representar a grupos de personas cuyos miembros muestran objetivos similares. Después, los miembros de los grupos lógicos se reparten por los diversos grupos físicos.

Una vez definidos estos grupos, la simulación emplea un tiempo determinado variable para realizar actualizaciones sobre el entorno y los individuos. Durante este tiempo, dos modelos distintos se calculan (uno para cada tipo de grupos). Las personas de los grupos físicos modifican su posición de tal forma que nuevos grupos físicos se forman mediante los individuos que compartían un comportamiento u objetivo similar. No obstante, si un grupo físico no tiene miembros es descartado. Finalmente, se obtienen grupos físicos de personas que coinciden con los grupos lógicos creados al principio.

Como se puede ver, en este caso se consideran individuos, mientras que algunas de sus características se heredan de los grupos a los que pertenecen. Ello permite

simplificar la especificación, ya que no es necesario describir todas las características de cada uno de los individuos.

2.2.4 Ingeniería del Software Orientada a Agentes

La ISOA agrupa una serie de metodologías para el desarrollo de software general donde el concepto principal es el de *agente*. Existe una enorme variedad de interpretaciones sobre el concepto de agente, pero en la mayoría de los casos se les suelen atribuir las siguientes características [96]:

- *Autonomía*. Los agentes encapsulan un estado propio que no es accesible desde el exterior y en el cual se basan para tomar decisiones sin intervención externa.
- *Reactividad*. Los agentes están situados en un entorno. Son capaces de percibir información procedente del mismo e interactuar con este modificándolo.
- *Proactividad*. Los agentes deben ser capaces no sólo de reaccionar frente a situaciones, sino de perseguir sus propios objetivos o metas mostrando iniciativa.
- *Habilidad social*. Los agentes interactúan con otros agentes. Suelen presentar habilidades sociales, tales como la negociación o la cooperación en la resolución de problemas.

Los agentes pueden agruparse formando Sistemas Multi-Agente (SMA) [93]. Ello les permite abordar la resolución de problemas que no pueden ser resueltos por un solo agente. Los SMA son muy apropiados para abordar sistemas distribuidos (tanto desde el punto de vista del modelado como de la implementación). Esto los hace recomendables en modelos microscópicos y mesoscópicos de simulación (ver Sección 2.2.3), donde considerar un número elevado de individuos puede requerir especificaciones complejas y necesita unos elevados recursos computacionales.

A continuación se introducen dos de las metodologías basadas en agentes más populares en la actualidad, Tropos (ver Sección 2.2.4.1) e INGENIAS (ver Sección 2.2.4.2). Estas ilustran los principales conceptos y algunos modelos de agentes usados en la ISOA.

2.2.4.1 Tropos

Tropos [9] es una metodología de desarrollo de SMA dirigida principalmente al análisis de requisitos siguiendo conceptos sociales e intencionales. El SMA se considera como una organización de *actores* (los *agentes*) con autonomía propia. El sistema se diseña como una estructura organizativa de actores donde estos satisfacen sus objetivos. Algunos de los conceptos básicos (ver Figura 5) manejados en estos modelos son:

- *Objetivo duro/Objetivo suave*. Estos objetivos representan los intereses de cada actor. Los objetivos duros se distinguen de los suaves en que para los primeros existe un criterio cuantitativo claro para definir si han sido satisfechos.
- *Plan*. Representa una forma de llevar a cabo una acción a nivel abstracto. La ejecución del plan puede ser un procedimiento para satisfacer un objetivo duro o suave.
- *Recurso*. Representa una entidad física o de información.

- *Capacidad*. Representa la capacidad de un actor de definir, elegir y ejecutar un plan para el cumplimiento de un objetivo, según ciertas condiciones del mundo y en presencia de un evento específico.
- *Creencia*. Representa el conocimiento que tiene el actor del mundo que le rodea.
- *Dependencia*. Es una relación intencional y estratégica entre dos actores. Ésta indica que un actor depende de otro actor para alcanzar un objetivo, llevar a cabo un plan u obtener un recurso.
- *Contribución*. Es una relación entre objetivos o planes y representa la forma en que estos pueden contribuir a la satisfacción de otro objetivo (de manera positiva o negativa).

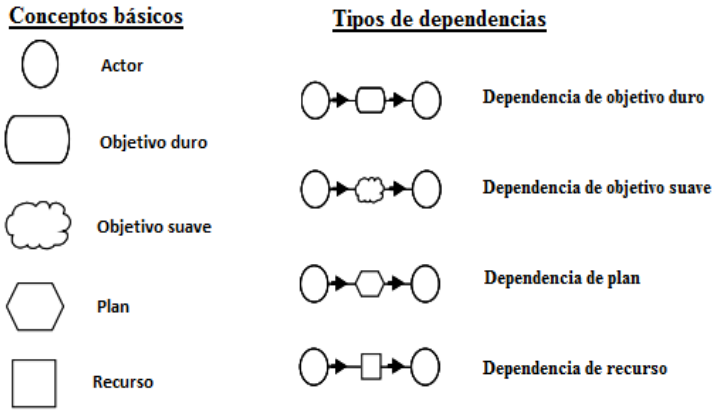

Figura 5: Conceptos y dependencias de Tropos extraídos de [71].

2.2.4.2 INGENIAS

INGENIAS [64] es una metodología de agentes que utiliza conceptos de la ingeniería del software, como la definición de flujos de trabajo, la encapsulación de funcionalidades a través de roles, o la distinción de diferentes puntos de vista en la descripción del sistema. Ha sido especialmente diseñada para construir SMA mediante especificaciones incrementalmente, a la vez que se garantiza la corrección del desarrollo. Tiene en cuenta el análisis, el diseño y las fases de implementación del ciclo de vida del desarrollo de software [63]. El curso del desarrollo en estas etapas se gestiona a través de variantes del Proceso Unificado de Desarrollo de Software (PUDS, *Unified Software Development Process*, USDP por sus siglas en inglés) o metodologías ágiles. Se automatiza el proceso de generación de código y documentación a partir de los modelos diseñados en cada una de las vistas mediante herramientas de ISDM.

INGENIAS define sus especificaciones usando un LM propio (algunos de sus elementos pueden verse en la Figura 6). Las especificaciones se descomponen en una serie de modelos:

- *Modelo de agente*. Describe los agentes individuales, sus tareas, metas, estado mental y roles.
- *Modelo de interacción*. Describe cómo se lleva a cabo la interacción entre los agentes. La declaración de cada interacción incluye los actores involucrados,

los objetivos perseguidos y una descripción del protocolo que sigue a la interacción.

- *Modelo de tareas y objetivos*. Describe las relaciones entre los objetivos y las tareas, y las estructuras de objetivos y tareas mediantes descomposiciones (por ejemplo, AND y OR) o dependencias (por ejemplo, precedencia).
- *Modelo de organización*. Describe cómo se agrupan los componentes (agentes, roles, recursos y aplicaciones) del sistema, qué tareas se ejecutan en común, que objetivos comparten, y qué limitaciones existen en la interacción entre los agentes.
- *Modelo de Entorno*. Define la percepción del agente en términos de elementos existentes del sistema. También identifica los recursos del sistema y quién es responsable de su gestión.

Los desarrolladores trabajan con estos modelos y en las tareas de generación de código usando la herramienta de desarrollo llamada Kit de Desarrollo de INGENIAS (IDK, *INGENIAS Development Kit*). El IDK incorpora un editor gráfico para especificaciones con el LM de INGENIAS y plug-ins para realizar transformaciones. Entre los plug-ins incluidos en la distribución de la herramienta están los usados para generar documentación HTML (*HyperText Markup Language*) y código fuente. Esta generación de texto se apoya en plantillas que usan un lenguaje específico de marcado para referirse a los elementos de las especificaciones.

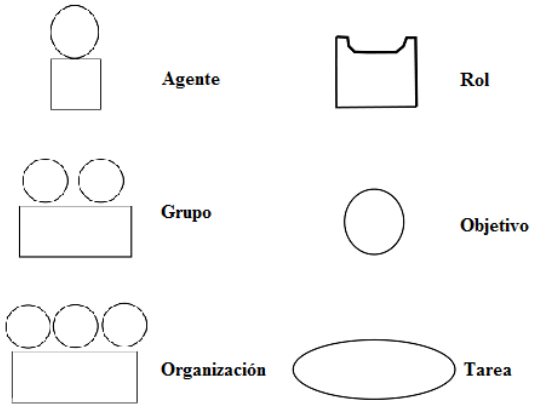

Figura 6: Representación de algunos conceptos de INGENIAS.

2.3 Procesos de desarrollo

Esta sección estudia las ventajas y desventajas de los modelos de desarrollo tradicionales y los modelos basados en la ISDM. Para ello se comparan trabajos desarrollados siguiendo ambas aproximaciones. Además se introducen los conceptos principales de la ISDM.

Las siguientes secciones detallan estos aspectos. La Sección 2.3.1 introduce los principales aspectos de la ISDM, en contraposición con los desarrollos tradicionales

centrados en el código. Después, la Sección 2.3.2 compara desarrollos tradicionales e ISDM aplicados a simulaciones de tráfico.

2.3.1 ISDM

La ISDM [22, 38] organiza el desarrollo de proyectos alrededor de los *modelos*. Estos modelos son compatibles con LM bien definidos. El proceso de desarrollo se centra en la producción de especificaciones iterativa e incrementalmente, con los desarrolladores refinando y añadiendo nuevos elementos en cada paso. Así se transita desde los modelos abstractos hasta los concretos, próximos a la plataforma de destino y el código. Durante este proceso se introducen *transformaciones* con el fin de automatizar las modificaciones repetitivas en los modelos. Por ejemplo, modificar modelos con la introducción de patrones o especializaciones concretas para una plataforma destino. Otros elementos del desarrollo (por ejemplo, código fuente o documentación) se generan de la misma forma, ya que estos pueden ser obtenidos a partir de los modelos utilizando transformaciones y ajustes manuales.

El *modelo* es una representación de un elemento (en este caso, un objeto, sistema o idea), de forma diferente al de la entidad misma. Éste captura aspectos esenciales del elemento representado para un cierto propósito, por ejemplo, desarrollar código o comprender un sistema existente.

De acuerdo con el Grupo de Gestión de Objetos (OMG, *Object Management Group*), la abstracción en el modelado tiene una serie de niveles jerárquicos en los que cada nivel inferior se apoya en los anteriores. Estos niveles son los siguientes [83]:

- *Nivel M3 o de Meta-metamodelo*. Es el nivel superior. Se consideran las características comunes de subconjuntos de modelos dando lugar a metamodelos que se agrupan mediante relaciones. El resultado final es un meta-metamodelo. En este nivel se encuentran los lenguajes de metamodelado usados para definir los elementos del siguiente nivel (ver Sección 2.3.1.1).
- *Nivel M2 o de Metamodelo*. Consiste en un conjunto de modelos o meta-metadatos (datos sobre los datos) que definen la estructura, la semántica y las características comunes de estos. Esto produce meta-metadatos o modelos que se agrupan mediante relaciones. El resultado final es un metamodelo. Se trata de la descripción de un LM.
- *Nivel M1 o de Modelo*. Se llevan a cabo agregaciones informales de metadatos que describen las características comunes de los datos produciendo en consecuencia agrupaciones mediante relaciones. El resultado final es un modelo.
- *Nivel M0 o de Información*. Agregación informal de datos a manejar en un entorno concreto (es decir, una aplicación). De esta manera se separa un subconjunto de datos con características comunes desde una perspectiva concreta.

Nótese que los niveles anteriores están organizados en base al concepto de metamodelo. Los metamodelos son una de las alternativas para definir LM, y son probablemente la más popular en la ISDM. No obstante, se consideran también otras alternativas, como las definiciones basadas en gramáticas para lenguajes textuales [8].

Por otra parte están las *transformaciones*, las cuales son procesos para convertir unos artefactos (por ejemplo, modelo o texto) en otros. Éstas pueden ser clasificadas de

acuerdo con la entrada y la salida que producen [14] en: Modelo a Modelo (M2M), Modelo a Texto (M2T), y Texto a Modelo (T2M).

Las transformaciones se implementan de diferentes maneras, incluyendo el uso de lenguajes de programación de propósito general y lenguajes de transformaciones. En el primer caso, la transformación se convierte en un módulo que utiliza interfaces de programación para manipular sus entradas y salidas. En el segundo caso, la transformación está escrita en un lenguaje específico para las transformaciones y existe un motor que la ejecuta. El primer enfoque tiene la ventaja de poder reutilizar artefactos ya probados y las herramientas de los principales enfoques de desarrollo (por ejemplo, programación orientada a objetos o procesamiento XML (*eXtensible Markup Language*)), y de facilitar el ajuste de la ejecución de la transformación. El segundo enfoque facilita la comprensión y el análisis de la correspondencia entre entradas y salidas.

Respecto a los conceptos de la ISDM utilizados en el marco de desarrollo de esta tesis destaca la organización por niveles. De ella se han considerado el nivel M2 (en el metamodelo que describe al LMT), el nivel M1 (en las especificaciones de modelos basadas en el LMT). En cuanto a las transformaciones, en este trabajo consisten fundamentalmente en la generación de código fuente partiendo de una especificación de modelo determinada (es decir M2T). Estas transformaciones se desarrollan con lenguajes estándares de programación usando un generador de código. Éste también proporciona librerías con las interfaces necesarias para manipular los modelos.

2.3.1.1. Lenguajes de metamodelado

Existen varias alternativas en lenguajes de metamodelado. Las más comunes están relacionadas con el estándar de OMG llamado MOF (*Meta-Object Facility*) [58].Éste se orienta a la definición de LM gráficos basados en grafos. Es la base usada para definir otros estándares de OMG como el Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*) [59] y el Metamodelo de Ingeniería de Procesos Software y de Sistemas (SPEM, *Software & Systems Process Engineering Metamodel*) [56]. MOF no cuenta sin embargo con demasiadas herramientas de soporte, lo que hace que sea una alternativa menos popular para el desarrollo.

Ecore es el lenguaje de metamodelado adoptado en los proyectos de Eclipse relacionados con la ISDM [27], por lo que cuenta con un amplio soporte en cuanto a herramientas de desarrollo. Además está prácticamente alineado con el subconjunto de MOF conocido como EMOF (*Essential MOF*).

Las principales primitivas de Ecore son (ver Figura 7):

- *EClass*. Agrupa elementos que comparten características.
- *EAttribute*. Es contenida por instancias de la anterior primitiva. Permite definir atributos con los tipos primitivos del lenguaje (*EDataTypes*), tales como enteros, booleanos o caracteres.
- *EReference*. Es una referencia binaria y dirigida que relaciona dos instancias de *EClass*. Permite crear dos tipos de referencias: *containment* y *non-containment*. Además soporta un mecanismo de herencia entre instancias de *EClass* mediante la referencia *ESuperType*.
- *EOperation*. Define un comportamiento determinado en una *EClass*.

- *EPackage*. Permite realizar agrupaciones entre elementos de un metamodelo. Pueden incluirse unos dentro de otros mediante un URI (*Uniform Resource Identifier*) que le identifica en el espacio de nombres.
- *EEnum.* Define tipos enumerados dentro de una lista *EEnumLiteral.*
- *EEnumLiteral*. Incluye los valores de un tipo enumerado denominándolos mediante nombres concretos.
- *EAnnotation*. Permite introducir anotaciones o comentarios haciendo referencia a alguna de las primitivas anteriores.

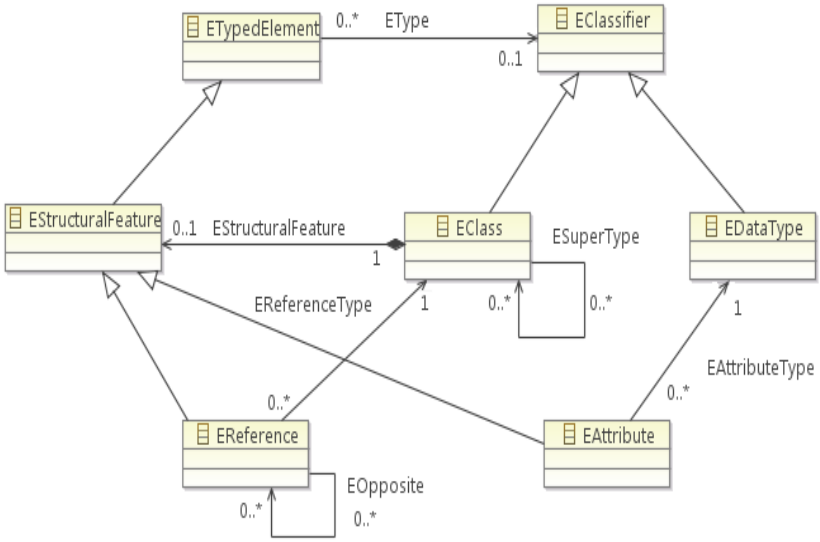

Figura 7: Extracto del modelo de Ecore.

Eclipse proporciona un amplio abanico de herramientas de soporte para el diseño de LM y sus modelos asociados. También permite la generación automática de código fuente mediante plantillas genéricas basadas en las metaclases propias del metamodelo. Estas funcionalidades se llevan a cabo mediante varios marcos de herramientas [27], siendo el núcleo el EMF (*Eclipse Modeling Framework*) [85] y el GEF (*Graphical Editing Framework*) [73]. El primero aborda principalmente el diseño de los LM, la transformación a código fuente y el desarrollo del editor gráfico correspondiente. El segundo permite implementar editores gráficos mediante plug-ins Eclipse partiendo de un metamodelo Ecore. Estos editores pueden ser configurados añadiéndolos propiedades a considerar mediante restricciones. Estas restricciones se especifican mediante el Lenguaje de Restricciones para Objetos (OCL, *Object Constraint Language*) [57].

Debido a esto, el marco de desarrollo de ISDM presentado en esta tesis se apoya en el lenguaje Ecore y las herramientas de Eclipse. El LMT es desarrollado mediante EMF mientras que el editor gráfico es un plug-in configurado por medio de GEF. Por su parte, el generador de código utiliza las plantillas de código producidas por EMF para

llevar a cabo las transformaciones automáticas a código fuente de las correspondientes especificaciones de modelos.

2.3.2 Modelos tradicionales y la ISDM

Los modelos tradicionales están basados en el código fuente y utilizan diversas aproximaciones para llevar a cabo su diseño e implementación (por ejemplo, el modelo en cascada o en espiral) [86]. La principal ventaja que presentan es la rapidez con la que puede llevarse a cabo el desarrollo de una aplicación en un entorno de trabajo homogéneo (es decir, desarrolladores con un mismo rol). No obstante, el mantenimiento y desarrollo de nuevos módulos de las aplicaciones basadas en la codificación son bastante costosos. Esto es debido a la poca flexibilidad que proporciona el código para su reutilización y evolución, en comparación con otros artefactos con un nivel de abstracción más alto. Además, la organización de los desarrolladores suele ser modular, por lo que suelen producirse problemas de desconocimiento parcial del código por parte de estos. Esto hace que el proceso de desarrollo se ralentice y proporcione un bajo umbral de reutilización. Para paliar estas situaciones se realiza documentación (incluyendo modelos) y se siguen guías de desarrollo, con el consecuente incremento de tiempo.

La mayor parte de los desarrollos de simulaciones se enmarcan dentro de estos modelos tradicionales. Por ejemplo, dos plataformas populares como MATSim [90] y SUMO [4] pertenecen a este grupo.

MATSim [90] es un kit multi-agente de herramientas de simulación basado en actividades e implementadas en Java. Este kit permite su extensión al ser de código abierto, y está diseñado para simular escenarios de gran escala [34]. La plataforma acepta múltiples representaciones del entorno, las cuales deben ser añadidas de manera independiente. No obstante, los archivos provenientes de OpenStreetMap [28] son perfectamente utilizables.

SUMO [4] es una plataforma de simulación de tráfico de código abierto. Su foco está en los vehículos, caracterizados principalmente a través de la ruta que siguen. Puede ser integrada con plataformas de agentes como Jade [6], adaptando su código fuente de forma que los elementos vehículo y peatón sean manejados por entidades con capacidades inteligentes. Por otra parte, es de las pocas plataformas que permite la simulación realista de intersecciones mediante la configuración de los semáforos.

En ambos casos, los artículos y tutoriales relacionados siempre abordan los elementos vía código o ficheros marcados, por lo que se trata de una aproximación tradicional al desarrollo.

Hay pocos ejemplos de enfoques de ISDM para simulaciones de tráfico, como el trabajo en [41]. Estos trabajos ilustran algunas de las ventajas ya mencionadas para los enfoques de ISDM, como la especificación a alto nivel de las simulaciones y hacer explícita toda la información relevante a través de modelos. Sin embargo, estos ejemplos con frecuencia no logran involucrar a algunos grupos de expertos. Tratan de cubrir el ciclo de desarrollo completo, desde la captura de los requisitos a la codificación, mientras que sus infraestructuras están orientadas típicamente sólo a algunas etapas. Por otra parte, utilizan formalismos y lenguajes que no son ampliamente utilizados en la comunidad de ISDM. El mencionado trabajo en [41] es un ejemplo de ello, al basar sus transformaciones en técnicas de rescritura de grafos.

2.4 Conclusiones

En este capítulo se ha revisado el contexto del trabajo actual. Se han analizado tanto los marcos conceptuales (en experimentos, modelos de especificación y plataformas de simulación) como los tipos de procesos de desarrollo asociados. De este análisis se pueden extraer varias conclusiones.

Los modelos basados en agentes parecen los más flexibles entre los contemplados en la literatura. Se pueden emplear como base para modelar otras propuestas como los autómatas. Además soportan modelos microscópicos y mesoscópicos. En el caso de estos últimos, es posible introducir características a nivel de grupo que afectan a los agentes individuales. Los mayores problemas aparecen con los modelos macroscópicos, donde no existe una manera intuitiva de abstraer, por ejemplo, ecuaciones de dinámicas como las usadas en [3].

En cuanto a los conceptos contemplados, en casi todos los casos se consideran atributos internos para definir el comportamiento de los participantes en el tráfico. Al menos estos implican un punto de origen y uno de destino, aunque estos puedan ser los de otro participante como en los modelos de seguimiento. Estos atributos pueden ser abstraídos como objetivos de los agentes. Existe además de necesidad de caracterizar otros aspectos de los participantes como la ansiedad o la agresividad.

Muchos modelos (por ejemplo, de agentes y autómatas celulares) consideran la influencia mutua entre los participantes en el tráfico. Por tanto, también se deben contemplar primitivas para este tipo de influencias.

Las metodologías de la ISOA tienen modelos ricos de agentes. Estos pueden cubrir parte de las necesidades señaladas anteriormente. Más importante, contemplan modelos de comportamiento interno del agente en función de sus características. Estos modelos pueden ser usados como modelos abstractos de comportamiento para los agentes de este trabajo.

Respecto a los procesos de desarrollo, existen pocos ejemplos en el campo de ISDM, y suelen usar infraestructuras propias. A fin de facilitar la adopción de estas aproximaciones y el mantenimiento de las herramientas, este trabajo adopta los proyectos de Eclipse para ISDM como su base. Estos son los de más amplia utilización en la comunidad de ISDM, y cumplen los requisitos para proporcionar el soporte necesario en este trabajo.

Capítulo 3. Lenguaje de modelado de tráfico

En este capítulo se introduce el Lenguaje de Modelado del Tráfico (LMT). Éste se define mediante un metamodelo siguiendo prácticas habituales de la ISDM. El capítulo presenta los distintos elementos (entidades, referencias, propiedades y restricciones) que forman parte del lenguaje, así como sus atributos y métodos. El lenguaje incluye varios elementos generales para las jerarquías de herencia y tres clústeres de conceptos que se usan para describir diferentes aspectos de las situaciones de tráfico.

3.1 Introducción

El LMT presentado en esta tesis está centrado en el modelado de los comportamientos de los individuos involucrados en el tráfico. Su objetivo principal consiste en integrar diferentes teorías relacionadas con el tráfico rodado para permitir el estudio apropiado de sus fenómenos en diferentes contextos. Para ello considera los múltiples roles que los individuos adoptan en el tráfico (es decir, conductores, peatones y pasajeros) y los que aparecen en los equipos de desarrollo (por ejemplo, experto en tráfico y programador). Se pretende que el LMT sea independiente de las plataformas de simulación, por lo que los detalles sobre éstas se consideran únicamente en las tareas de diseño finales.

Siguiendo las prácticas comunes en ISDM, se utiliza un metamodelo para describir el LM. Este metamodelo dispone de una serie de metaclases que representan varios de los conceptos mentales usado habitualmente en el campo de la ISOA [69, 93]. Como se indicó anteriormente el lenguaje de metamodelado usado es Ecore (ver Sección 2.3.1.1 y Capítulo 1 en general).

Con el fin de facilitar la extensión y especialización del LM para acomodar diferentes teorías y marcos conceptuales, su organización se basa en relaciones de herencia y composición. La herencia proporciona especialización en los conceptos y relaciones, mientras que las composiciones permiten detallar conceptos mediante relaciones entre grupos funcionales, enlaces físicos, partes o similitud.

El LM también pretende servir de guía a los expertos en su uso resaltando los grupos de elementos relacionados que se emplean para especificar cada parte de un problema. Con este propósito, el metamodelo se organiza en tres *clústeres* o grupos de conceptos: un clúster *Mental* donde se consideran las diferentes características de los individuos; un clúster *Entorno* para especificar la información del entorno de las personas, y un clúster *Interactivo* que representa las interacciones entre los individuos y el entorno, y la toma de decisiones y acción asociadas a las mismas.

El clúster *Mental* considera las características de los individuos, incluyendo la información interna que manejan. Desempeña un papel similar al *estado mental* en

numerosas propuestas del paradigma de agentes [81]. De hecho, incorpora algunos de sus conceptos para modelar la información actual que un individuo o grupo posee, como el conocimiento (metaclase *Knowledge*) y sus componentes (metaclase *KComponent*).

El clúster *Entorno* se basa en el enfoque CVE [1]. Este enfoque considera que los individuos involucrados en el tráfico pueden interactuar entre ellos o con su entorno, ya sea directamente o mediante los medios de transporte que utilicen. Estas interacciones son dinámicas e influyen en sus comportamientos individuales. Este supuesto se ajusta al MBA [2, 35], donde los agentes son entidades intencionales que pueden establecer comunicación entre ellos para diferentes propósitos (por ejemplo, colaborar o interactuar).

El clúster *Interactivo* modela la toma de decisiones y las interacciones de los individuos. Adapta este aspecto de metodologías ISOA como Tropos [9] (ver Sección 2.2.4.1) o INGENIAS [64] (ver Sección 2.2.4.2). La metaclase *Goal* representa los objetivos de las personas, mientras que la metaclase *Task* proporciona las instrucciones a ejecutar con el fin de satisfacer estos objetivos. Estos elementos son considerados dentro de un ciclo de percepción, razonamiento y actuación similar a los encontrados en la literatura sobre agentes inteligentes [43].

El resto del capítulo se organiza como sigue. La Sección 3.2 describe los conceptos en los que se basa el metamodelo. Por su parte la Sección 3.3 presenta las conclusiones sobre la estructura y funcionalidad del metamodelo en su conjunto.

3.2 Conceptos

El metamodelo propuesto ha sido diseñado teniendo en mente facilitar la comprensión de los problemas de tráfico y la adaptación de su LM a nuevas necesidades. Para alcanzar estos objetivos, el metamodelo propuesto adopta un enfoque de MBA, organizándose a través de la agrupación de conceptos, y mediante jerarquías de composición y herencia.

Los conceptos del metamodelo están estrechamente relacionados con el paradigma de agentes [81], y en particular con modelos donde se considera el estado mental [9, 64, 69] (ver Figura 8 para el modelo CDI [69]). Así, la metaclase *Knowledge* engloba las *creencias* y las *intenciones* de los individuos, mientras que la metaclase *KComponent* se encarga de organizar los posibles tipos de conocimientos (por ejemplo, la experiencia en la conducción o la familiaridad con la ruta planificada). Las características de los individuos son representadas por la metaclase *Profile*, y los tipos de características por la metaclase *KComponent* (por ejemplo, la edad o su grado de estrés). Los objetivos son representados por la metaclase *Goal*. Por su parte, la información del entorno se considera mediante la metaclase *Environment*, cuyos componentes se especifican mediante la metaclase *EComponent* (por ejemplo, señales de tráfico o las condiciones ambientales).

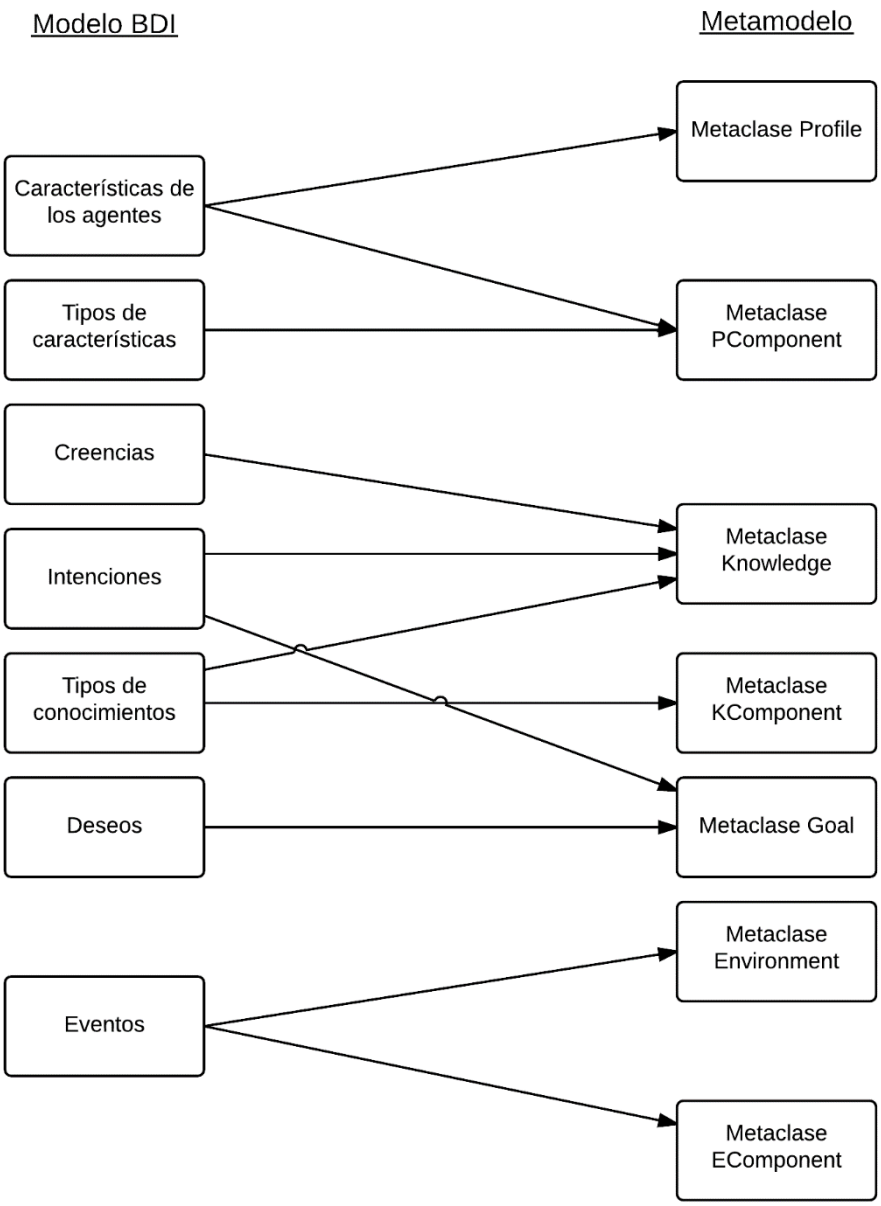

Figura 8: Equivalencias entre el metamodelo y el modelo CDI.

En cuanto a la estructura de los conceptos, estos se clasifican en tres clústeres. Los clústeres *Mental* y *Entorno* se encargan de describir los diferentes conceptos obtenidos de la literatura sobre tráfico. Ambos clústeres comparten estructuras similares. El clúster *Mental* considera el estado interno de los participantes en el tráfico [79], mientras que el clúster *Entorno* incluye el enfoque CVE, centrándose en las distintas formas de interacción entre los individuos involucrados en el tráfico y el entorno que los rodea. Por su parte, el clúster *Interactivo* se centra en la representación de los objetivos y acciones de las personas implicadas en el tráfico y su uso. Para ello se sigue un enfoque inspirado en las metodologías de la ISOA que integra un ciclo de percepción, razonamiento y actuación.

El elemento central del metamodelo es la metaclase *Person*. Ésta representa el tipo de personas que participan en el tráfico. De acuerdo con sus medios de transporte, estas personas pueden tomar el rol de conductores, pasajeros o peatones. Las instancias de *Person* pueden interactuar con una instancia de *Environment*. Esta interacción es directa (en el caso de los peatones mediante una instancia de la referencia *Perceives*), o indirecta (para los conductores y pasajeros mediante instancias de la referencia *Interacts*), ya que se usa un vehículo para ello. Sus objetivos son descritos por instancias de *Goal*, y las formas posibles para lograrlos están representadas por las instancias de *Task*. Las instancias de *Evaluator* interpretan la información disponible (quizás generando nueva información) y determinan cómo tiene que actuar la gente de acuerdo con las circunstancias observadas (es decir, las potenciales tareas que puede ejecutar en unas circunstancias dadas). Por último, las instancias de *Actuator* ejecutan las tareas planificadas.

Los elementos anteriores se organizan en jerarquías de herencia. Estas jerarquías proporcionan la especialización de conceptos y estructura necesaria al metamodelo. Todos los conceptos heredan de la metaclase *GeneralElement* (ver Figura 9). Esta metaclase proporciona la referencia *EInherits* para representar herencia entre elementos del mismo tipo en las especificaciones de modelos. Por su parte, la metaclase *GeneralRelationship* (ver Figura 9) soporta la introducción de las relaciones (por ejemplo, afecta o influye) entre otros elementos. La referencia *RInherits* permite su especialización. Ambos tipos de referencias son limitadas por restricciones OCL. Por ejemplo, estas restricciones sólo permiten herencia entre instancias del mismo tipo de metaclases (en el caso de una instancia de *Knowledge*, ésta sólo puede extenderse usando una instancia de la referencia *EInherits* a otra instancia de *Knowledge*).

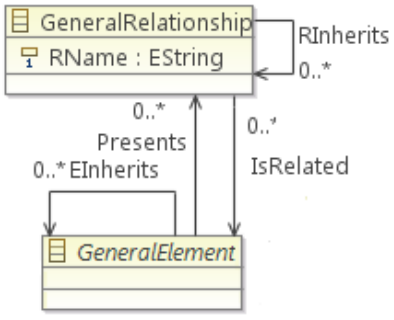

Figura 9: Estructura básica del metamodelo.

Las metaclases también incluyen atributos y métodos predefinidos. Los primeros pueden ser específicos de una metaclase, (como *AvailableArea* en la metaclase *Environment*), o comunes, con nombre y significado similar (como los atributos *XName*, por ejemplo *KName* o *VName*). No obstante, indicar el caso particular del atributo *Id* de la metaclase *Person*, el cual tiene un significado similar a los atributos *XName*, aunque los nombres son diferentes. Por otra parte, los métodos son contenedores para especificaciones que describen comportamiento o como derivar ciertos atributos de otros. Es decir, en la especificación de modelo se pueden insertar fragmentos de especificación (por ejemplo código o fórmulas) en el cuerpo de estos métodos. Dichas especificaciones pueden ser utilizadas más tarde para guiar transformaciones, por ejemplo durante la generación de código fuente.

La estructura interna de los clústeres *Mental* y *Entorno* permite jerarquías de composición utilizando las metaclases *XComponent* (por ejemplo, *VComponent* o *KComponent*). Estas metaclases pueden descomponerse en otras de su mismo tipo. Todas estas composiciones están limitadas por restricciones OCL. Por ejemplo, una instancia de *Profile* puede ser descompuesta solamente en instancias de *PComponent* mientras que estas instancias de *PComponent* sólo pueden ser descompuestas en otras del mismo tipo.

La Sección 3.2.1 describe el clúster *Mental* dedicado al estado mental y las características de los participantes en el tráfico. La Sección 3.2.2 se centra en el clúster *Entorno* con los conceptos para describir la configuración de tráfico de acuerdo con el modelo CVE [1]. Finalmente, la Sección 3.2.3 introduce el clúster *Interactivo* que incluye los conceptos para representar las interacciones entre los elementos anteriores y en la toma de decisiones.

3.2.1 Clúster *Mental*

El clúster *Mental* (ver Figura 10) representa los diferentes conceptos que pueden aparecer en el dominio del tráfico rodado y que pueden influir en el comportamiento de los individuos. Estos conceptos se clasifican como características de las personas o estado actual de las mismas.

El clúster está formado por tres metaclases principales: *Person*, *Profile* y *Knowledge*. *Profile* representa las diferentes características de las personas involucradas en el tráfico (por ejemplo, la edad o la fatiga). Por su parte, *Knowledge* considera el estado mental y el conocimiento actual (excepto los objetivos de la persona), y los planes que los individuos presentan cuando intervienen y forman parte del tráfico. Esta información es almacenada por el atributo *Facts*, pudiendo ser clasificada mediante un orden preestablecido. El conocimiento puede ser factual (por ejemplo, señales de tráfico), procedimental (por ejemplo, la forma de adelantar a un vehículo), y normativo (por ejemplo, los conductores deben respetar las distancias de seguridad con otros vehículos). Los planes de los individuos respecto al tráfico vienen dados frecuentemente por la ruta a seguir. Esta ruta es tenida en cuenta por el atributo *Route*, mientras que el progreso en la ruta realizado por los individuos viene dado por el atributo *RoutePlace*.

Tanto los atributos del conocimiento de las personas como de las características de las mismas pueden especificar información que no cambia en el tiempo que dura la simulación (por ejemplo, el género de una persona o el significado de las señales), o

que sí lo hace (por ejemplo, el nivel de estrés o el estado de ánimo). Para contemplar estos aspectos, el metamodelo proporciona los correspondientes métodos *calculateXValue* y atributos *XValues* asociados a cada una de las metaclases.

Las instancias de la metaclase *Knowledge* y sus metaclases de composición *KComponent* pueden representar información perteneciente a los individuos, o global disponible para todos los participantes en la simulación. Por ejemplo, los límites de velocidad son comunes a un modelo, pero algunos participantes pueden ignorarlos. El atributo *KIsGeneral* diferencia ambos usos.

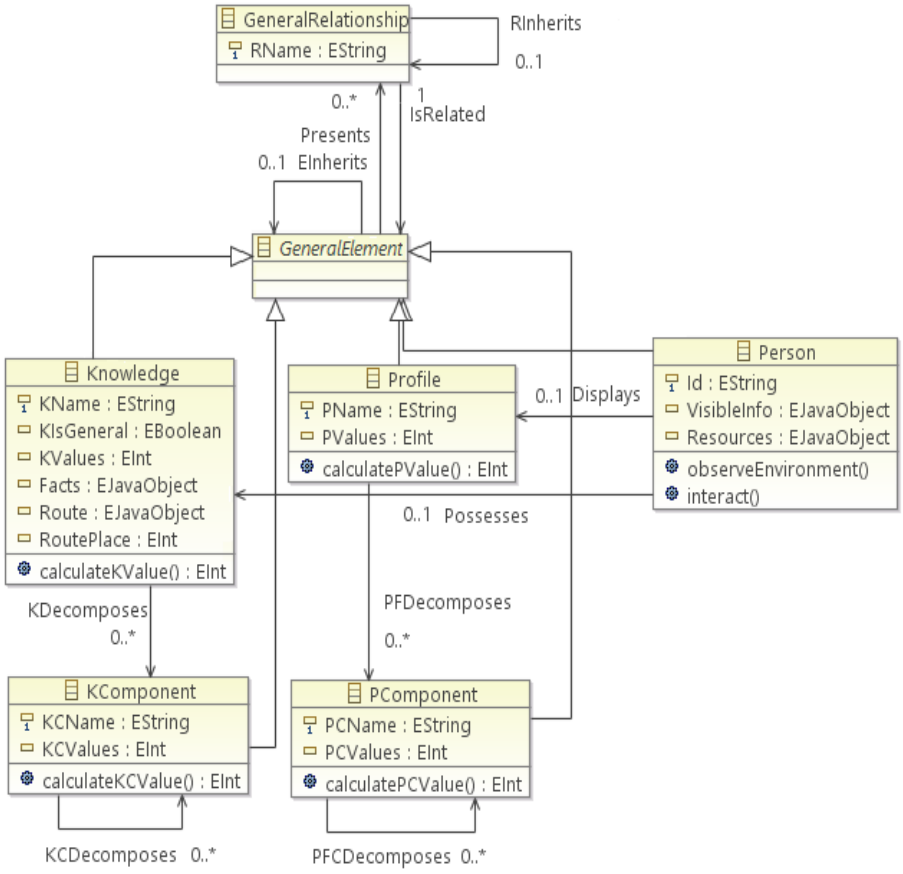

Figura 10: Clúster *Mental*.

3.2.2 Clúster *Entorno*

El tráfico se produce en un entorno que establece ciertas condiciones físicas (por ejemplo, tiempo, ancho de carril, u obstáculos). El clúster *Entorno* (ver Figura 11) considera estos aspectos mediante la adaptación de los conceptos del modelo CVE, y centrándose en el rol que juegan los conductores, los peatones y los acompañantes en el tráfico rodado. Para ello supone que un individuo puede obtener información del entorno (el caso de cualquier persona que participa en el tráfico) y del vehículo en el

que se encuentra (sólo los conductores y pasajeros). Estas consideraciones y sus elementos relacionados pueden ser extendidos para permitir acomodar otras teorías similares al CVE, por ejemplo modelos de autómatas reactivos para tráfico como [18] o consideraciones sobre factores externos que incrementan el riesgo de accidente [17].

El clúster se encarga de describir cómo las personas se relacionan con sus medios de transporte y con el entorno que las rodea. Para llevar a cabo estas descripciones, utiliza tres metaclases principales: *Person*, *Environment*, y *Vehicle*.

Person ya fue introducida a propósito del clúster *Mental* (ver Sección 3.2.1). Permite tener en cuenta los diferentes roles que pueden jugar las personas en el tráfico por carretera (es decir, conductor, pasajero o peatón). En el caso de los conductores se utiliza la referencia *Drives* para relacionar instancias de esta metaclase con instancias de la metaclase *Vehicle*, mientras que para los pasajeros se emplea la referencia *Uses*. Los peatones no utilizan ninguna de estas relaciones, ya que no se relacionan con ningún medio de transporte.

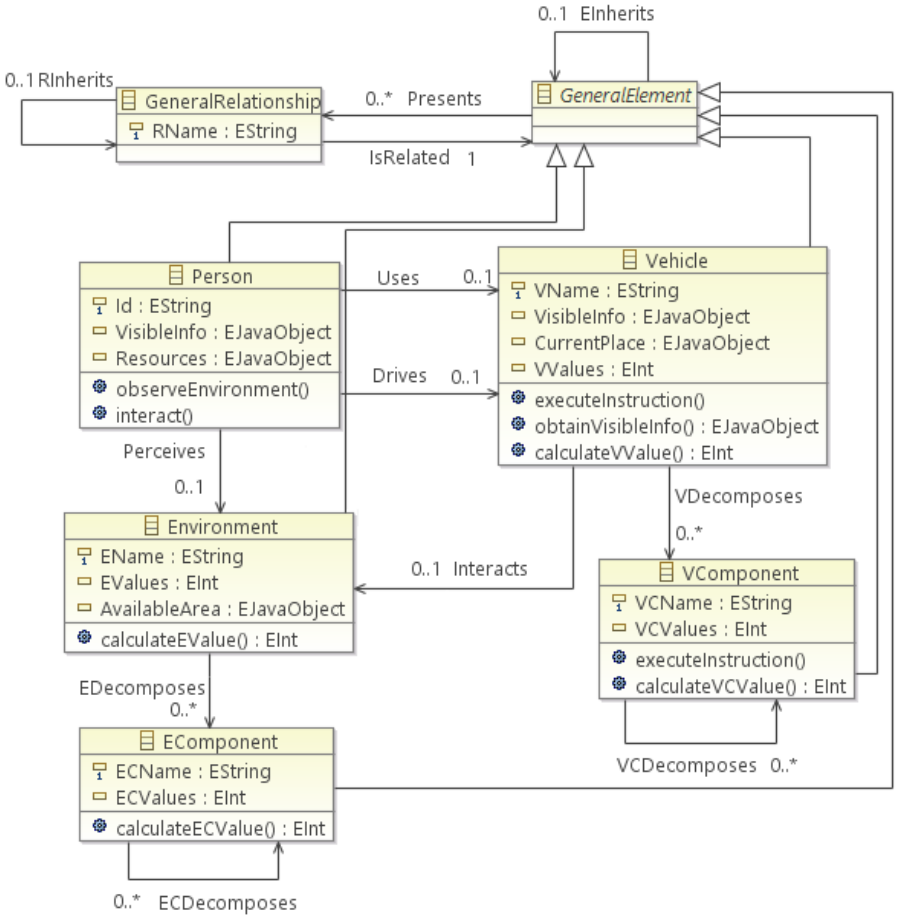

Figura 11: Clúster *Entorno*.

La metaclase *Environment* representa el lugar donde la gente (es decir, las instancias de la metaclase *Person*) interactúa, incluyendo las condiciones físicas que pueden ocurrir (por ejemplo, el clima y las características del camino). Estas condiciones son consideradas por instancias de la metaclase *EComponent*. Una especificación de modelo conforme al LMT puede tener a lo sumo una instancia *Environment* que comparten todos los demás elementos de la especificación.

La metaclase *Vehicle* simboliza el medio de transporte que utiliza cada individuo. Los conductores y sus pasajeros se relacionan con las instancias de *Environment* a través de sus vehículos, pero sólo los conductores pueden realizar acciones sobre ellos para actuar sobre el vehículo y el entorno. En el caso de los peatones, presentan una relación directa con el entorno. Mediante la metaclase *VComponent* el LMT facilita la inserción de características propias del medio de transporte (por ejemplo, el tamaño o la cilindrada del vehículo).

Las influencias mutuas entre las instancias de *Person*, *Environment* y *Vehicle* a través de sus relaciones están representadas parcialmente en el metamodelo mediante algunos atributos y métodos. La metaclase *Environment* tiene un atributo *AvailableArea* que indica la parte del entorno que puede ser percibida. Las metaclases *Person* y *Vehicle* incluyen el atributo *VisibleInfo* para especificar qué información del atributo *AvailableArea* de la instancia *Environment* se puede percibir. La metaclase *Person* dispone de un método *observeEnvironment* para poder actualizar la percepción del entorno modificando los atributos *VisibleInfo* de sus propias instancias. También presenta un método *interact* para llevar a cabo las interacciones con el entorno y con el resto de individuos. La metaclase *Vehicle* presenta un método llamado *obtainVisibleInfo* encargado de realizar una tarea similar al método *observeEnvironment* de la metaclase *Person*. Además, esta metaclase cuenta con el método *executeInstruction* destinado a realizar tareas específicas del vehículo (por ejemplo, mover un espejo retrovisor o encender las luces). La metaclase *VComponent* presenta este mismo método con una funcionalidad análoga.

3.2.3 Clúster *Interactivo*

El clúster *Interactivo* (ver Figura 12) describe cómo las instancias de *Person* actúan sobre las situaciones de tráfico descritas con los elementos de los clústeres *Mental* (ver Figura 10) y *Entorno* (ver Figura 11). Sus componentes están organizados en dos grupos. El primero de ellos describe los objetivos de las personas y sus capacidades para intentar alcanzarlos. El segundo representa los elementos que llevan a cabo el ciclo de percepción, razonamiento y actuación.

El primer grupo incluye las metaclases *Goal* y *Task*. Estos dos conceptos se adoptan de la ISOA, donde metodologías basadas en agentes como Tropos [9] e INGENIAS [64] los usan para especificar los SMA [93] (ver Sección 2.2.4). Estas metodologías incluyen una arquitectura específica de actuación donde los agentes desempeñan múltiples roles que les confieren distintas habilidades, objetivos y conocimientos. Los agentes tratan de hacer cumplir las condiciones de satisfacción de sus diferentes objetivos. Para ello, los agentes tienen tareas vinculadas a los objetivos, de tal manera que la ejecución de las tareas es capaz potencialmente de satisfacer los objetivos. Estas tareas son habilitadas como ejecutables de acuerdo con condiciones que dependen del estado mental de sus agentes y del entorno. Tanto las tareas como los objetivos pueden

descomponerse en otras tareas u objetivos respectivamente, generando composiciones *AND* y *OR*.

En este caso, el clúster *Mental* representa el estado mental de los agentes, y el clúster *Entorno* proporciona la información procedente del entorno y el vehículo (sólo para el caso de los conductores y pasajeros). La metaclase *Goal* representa un estado de algunos elementos de la situación de tráfico que una persona aspira a mantener o alcanzar, mientras que la metaclase *Task* modela las habilidades de la persona. Ambas metaclases disponen de atributos específicos para caracterizar estos aspectos. La metaclase *Goal* tiene un atributo *Satisfaction* que representa sus condiciones de satisfacción. La metaclase *Task* incluye un atributo *Instructions* para especificar las acciones atómicas que se llevan a cabo para realizarla. Estas acciones se ejecutan para obtener la satisfacción del objetivo al que están asociadas las tareas mediante una instancia de la referencia *GImplies*.

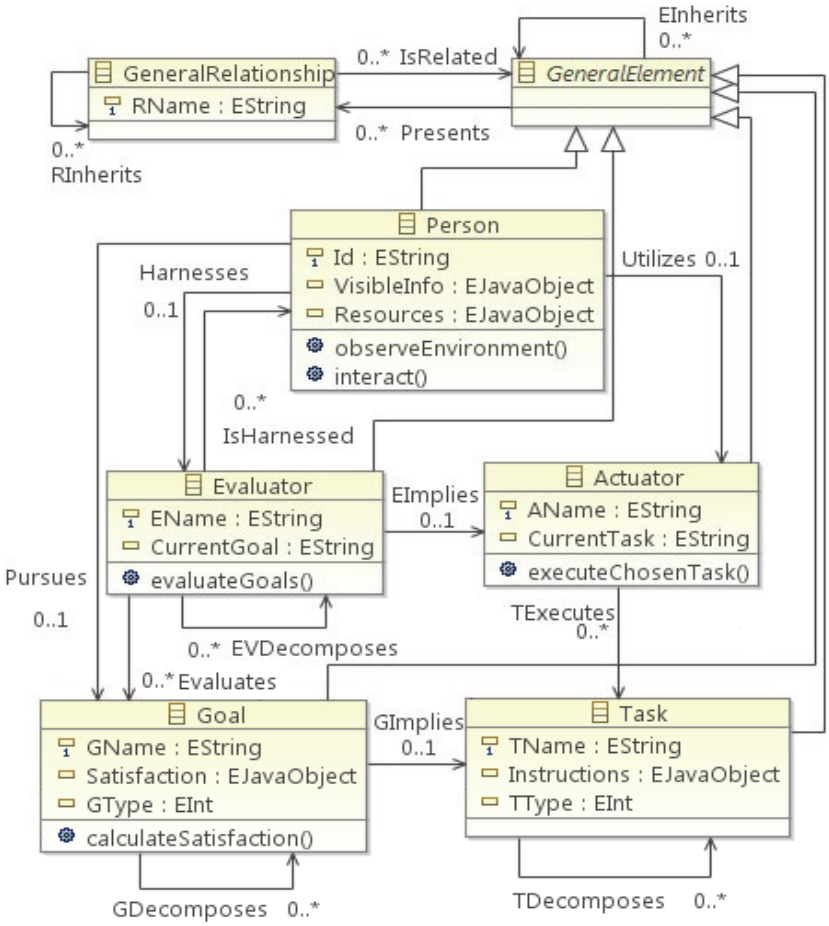

Figura 12: Clúster *Interactivo*.

Estas metaclases pueden descomponerse en otras de su tipo (restringidas mediante expresiones OCL), siguiendo la estructura ya vista para los *XComponent* de los otros dos clústeres. Sin embargo, su semántica es diferente, siendo similar a las composiciones de tareas y objetivos en las metodologías basadas en agentes previamente introducidas (ver Sección 2.2.4). En este caso su semántica se relaciona con la satisfacción en lugar de determinar las características de un componente. Las metaclases *Goal* y *Task* disponen de atributos *GType* y *TType* con el fin de especificar el tipo de composiciones (por ejemplo, *AND* u *OR*). El atributo *GType* representa las composiciones de satisfacción del objetivo, mientras que el atributo *TType* indica si la tarea actual es llevada a cabo completando una o la totalidad de sus sub-tareas. Esta semántica es flexible, en el sentido de que ambos atributos pueden soportar estructuras y clasificaciones adicionales.

El segundo grupo presenta los elementos de una instancia de *Person* que están encargados de evaluar el estado conocido (del entorno e interno del agente) y ejecutar acciones. Este grupo sigue un ciclo clásico de percepción, razonamiento y actuación para agentes [43]. En él, la información percibida del entorno se almacena en los elementos del clúster *Entorno* (incluyendo la metaclase *Person*), el razonamiento se lleva a cabo por las instancias de la metaclase *Evaluator*, y la actuación se realiza por medio de instancias de la metaclase *Actuator*.

Las instancias de *Evaluator* pueden descomponerse en otras mediante la referencia *EVDecomposes* (ver Figura 12), permitiendo distribuir las responsabilidades entre ellas. En el caso de las instancias de *Actuator*, éstas no se pueden descomponer en otras. Cada instancia de *Person* (es decir, un tipo de persona modelada) sólo puede presentar una instancia de *Actuator* con la que se relaciona mediante la referencia *Utilizes* (ver Figura 12). Las instancias de ambas metaclases pueden refinarse utilizando la herencia a través de las referencias *EInherits*.

En cuanto al funcionamiento del clúster, las instancias de *Evaluator* evalúan la información obtenida de las instancias *Environment*, *Vehicle*, *Profile* y *Knowledge* vinculadas a su instancia *Person*, de sus componentes, y de otros posible elementos relacionados a través de instancias de *GeneralRelationship*. Para llevar a cabo este proceso estas instancias *Evaluator* disponen del método predefinido *evaluateGoals*. Con ello actualizan el estado interno de su instancia de *Person*. Toda esta información determina el estado actual de las instancias de *Goal*, es decir, si se satisfacen o no. Para ello, estas instancias disponen del método *calculateSatisfaction*. Una vez que se selecciona una instancia candidata de *Goal* para su ejecución, una instancia de *Actuator* recopila sus tareas asociadas. Estas instancias de *Task* son ejecutadas mediante el método *executeChosenTask* siguiendo las instrucciones atómicas recogidas en sus atributos *Instructions* o ejecutando sus instancias de *Task* hijas.

3.3 Conclusiones

En este capítulo de tesis se ha presentado un metamodelo extensible que describe el LMT para apoyar un enfoque de ISDM para el desarrollo de simulaciones de tráfico en múltiples plataformas destino. El LM está centrado en el comportamiento de los

individuos involucrados en el tráfico y en los roles que estos desempeñan (conductores, peatones y pasajeros).

El LMT se ha diseñado para ser capaz de integrar diferentes teorías existentes en el dominio. Para ello sigue un enfoque basado en el modelado de agentes con el fin de tener en cuenta aspectos de percepción, razonamiento y toma de decisiones de personas, y las interacciones sociales de los individuos involucrados en el tráfico rodado.

El metamodelo consta de tres clústeres que agrupan sus conceptos: *Mental*, *Entorno* e *Interactivo*. El primero incluye los diferentes aspectos personales de cada individuo que pueden influir en su comportamiento. Estos aspectos se clasifican en dos grupos: las características propias de cada persona o grupo de personas, y su conocimiento o estado actual. Este clúster desempeña un papel similar al estado mental del paradigma de agentes [81]. El segundo clúster se basa en el enfoque CVE [1] para modelar las diferentes interacciones de los individuos involucrados en el tráfico rodado, teniendo en cuenta las relaciones entre los vehículos, el entorno y las personas. Esta estructura favorece la adaptación de los conceptos de muchos de los estudios existentes, ya que incluye nociones ampliamente aceptadas para describir las situaciones de tráfico. El último clúster utiliza conceptos como *Goal* y *Task* procedentes de la ISOA, e integra un ciclo de percepción, razonamiento y actuación mediante los conceptos *Evaluator* y *Actuator*.

Las metaclases del metamodelo están diseñadas para soportar estructuras jerárquicas (por ejemplo, aquellas que incluye como contenedor la metaclase *Profile* y sus componentes con la metaclase *PComponent*). La herencia entre elementos del mismo tipo se introduce para hacer posible las especializaciones de los mismos. Otros tipos de relaciones entre elementos también son considerados.

El LMT presentado en esta tesis no considera aún ciertos aspectos que aparecen en algunas propuestas del dominio. Por ejemplo, no contempla las normas sociales existentes en el tráfico [72]. Posibilita introducir las características de los vehículos presentes en el tráfico rodado (mediante la metaclase *VComponent*), pero no permite diferenciar entre sus distintos tipos (por ejemplo, camiones [13] o motocicletas [29]). Además, no facilita la representación de la influencia de las señales de tráfico (por ejemplo, cruces o semáforos) sobre los sujetos de manera individual. Esta representación puede llevarse a cabo a nivel colectivo por medio de las metaclases *GeneralRelationship* y *EComponent.* Así, las primeras pueden denotar esta influencia sobre los individuos del tipo indicado por las instancias de la metaclase *Person*, mientras que las últimas pueden especificar los diferentes tipos de señales contempladas.

Capítulo 4. Herramientas de desarrollo

Este capítulo introduce las distintas herramientas de desarrollo que forman parte de la infraestructura de ISDM presentada en esta tesis. La misión de estas herramientas es facilitar a los usuarios la realización del proceso de desarrollo, automatizando en lo posible la mayor cantidad de tareas repetitivas y permitiéndoles centrarse en aquellas que requieren mayor conocimiento. Para ello, estas herramientas toman como base el metamodelo anteriormente presentado y las plantillas de código que Eclipse genera automáticamente a partir de él. Las tareas que soportan incluyen la especificación de modelos conformes al LMT y la introducción de restricciones sobre ellos, así como generar transformaciones semiautomáticas de modelo a código fuente adaptadas a una plataforma de simulación de tráfico determinada. Además, una plataforma de test ha sido implementada para llevar a cabo diversas pruebas. El capítulo finaliza con una revisión de las características de las aplicaciones y las ventajas y limitaciones que su implementación reporta.

4.1 Introducción

Este capítulo presenta las distintas herramientas de desarrollo de la infraestructura de ISDM de esta tesis. Estas herramientas incluyen un editor gráfico de modelos, un generador de código y una plataforma de test.

El editor gráfico se desarrolla mediante distintas herramientas que aporta GEF [73]. Proporciona un lienzo y una paleta para el diseño gráfico de las especificaciones de modelos. Para ello utiliza como punto de partida el metamodelo diseñado mediante EMF [85] descrito anteriormente (ver Capítulo 3).

El generador de código se ha implementado para facilitar la realización de trasformaciones de modelos a código fuente. Permite la reutilización de las especificaciones creadas mediante el editor gráfico. Soporta la creación de las transformaciones a través de diversas funcionalidades gestionadas mediante asistentes. Puede generar dos tipos de archivos finales: un plug-in directamente insertable en la plataforma destino, o un fichero comprimido ejecutable que empaqueta la antigua plataforma, la especificación de modelo y todo el código de la simulación específico para la plataforma.

La plataforma de test ha sido desarrollada para realizar pruebas con distintas configuraciones. Esta plataforma utiliza un SMA basado en la plataforma de agentes A-Globe [82]. El SMA implementa los individuos involucrados en el tráfico, generando los diferentes comportamientos requeridos.

La Sección 4.2 describe el editor gráfico usado para realizar especificaciones de modelo, la Sección 4.3 introduce el generador de código con sus distintas

funcionalidades y la Sección 4.4 presenta la plataforma de test desarrollada. Finalmente, la Sección 4.5 recoge las conclusiones obtenidas y muestra las ventajas y desventajas de la utilización de estas herramientas.

4.2 Editor gráfico

El editor gráfico es un plug-in de Eclipse que guía a los usuarios en el desarrollo de las especificaciones de modelos. Genera especificaciones compatibles con el LMT mediante dos ficheros con formato XMI (*XML Metadata Interchage*) [60]: uno para almacenar el diseño de la especificación de modelo y otro para la estructura gráfica de la misma. El primero permite la validación de la especificación de modelo creada, asegurando su conformidad con el LMT y las restricciones OCL añadidas (ver Figura 13). El segundo permite visualizar gráficamente la especificación mediante un lienzo y una paleta. El lienzo permite mostrar el estado actual del diseño creado, mientras que la paleta proporciona los mecanismos adecuados para la instanciación de los conceptos del metamodelo disponibles para la especificación de modelo (ver Figura 14).

Está herramienta se construye a partir del metamodelo introducido en el capítulo anterior. Para ello utiliza los archivos generados por EMF [85] durante la creación del metamodelo y el código fuente que produce automáticamente por defecto.

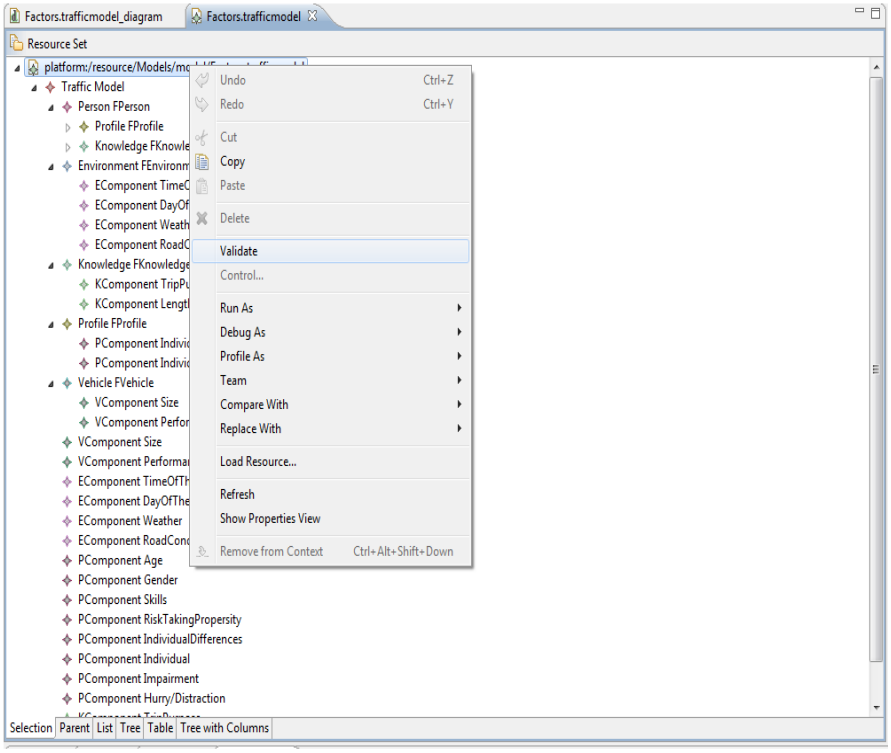

Figura 13: Validación del diseño de la especificación de modelo.

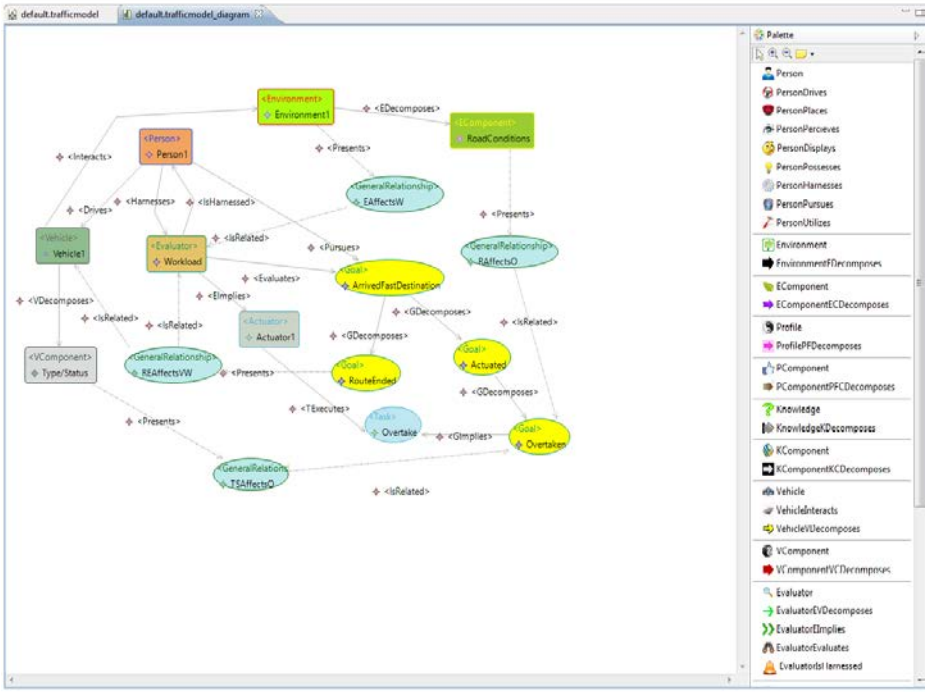

Figura 14: Lienzo y paleta de elementos del editor gráfico.

Por su parte, los archivos de configuración necesarios para el funcionamiento del plug-in (ver Figura 15) son desarrollados mediante GEF [73]. Estos archivos son los siguientes:

- El archivo con extensión *gmftool*. Se encarga de generar la paleta de elementos disponibles. Permite la inserción de imágenes con el objetivo de hacer más amigable la interfaz y separadores entre los elementos para formar grupos.
- El archivo con extensión *gmfgraph*. Se ocupa de los elementos del lienzo. Permite la configuración de estos indicando una representación identificativa y el color. Además proporciona la opción de mostrar o no atributos del elemento (por ejemplo, el nombre).
- El archivo con extensión *gmfmap*. Lleva a cabo la integración de los dos archivos anteriormente introducidos indicando qué elemento del lienzo es creado por la paleta de elementos. Permite la inserción de restricciones OCL para guiar al usuario en el desarrollo de las especificaciones de modelos. En este caso se introduce una restricción en la referencia *EInherits* para permitir que el resto de elementos del metamodelo solo puedan heredar de elementos del mismo tipo (ver Figura 16), y otra restricción en la referencia *RInherits* para que las instancias de la metaclase *GeneralRelationship* solo puedan heredar de elementos de este tipo (ver Figura 17).
- El archivo con extensión *gmfgen*. Es creado a partir del anterior y automatiza el proceso de la creación del plug-in basándose en la información contenida en el

resto de archivos. Genera una carpeta con extensión *diagram*. Esta carpeta contiene un editor ejecutable como aplicación Eclipse. Cuando se lleva a cabo esta acción, se despliega el editor gráfico en una nueva ventana.

Finalmente, indicar que el funcionamiento del editor gráfico también requiere la presencia de las carpetas con extensiones *edit* y *editor*. Estas se crean para el desarrollo del metamodelo con EMF [85].

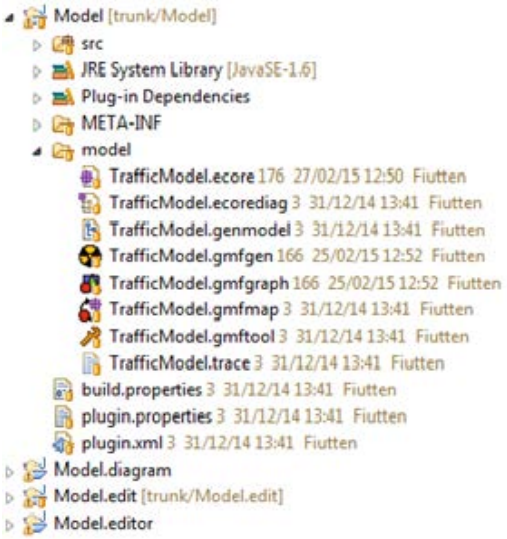

Figura 15: Archivos necesarios para la generación del editor gráfico.

```
(self <> oppositeEnd) and ((self.oclIsTypeOf(Person) and oppositeEnd.oclIsTypeOf(Person)) or
(self.oclIsTypeOf(Knowledge) and oppositeEnd.oclIsTypeOf(Knowledge)) or
(self.oclIsTypeOf(Profile) and oppositeEnd.oclIsTypeOf(Profile)) or
(self.oclIsTypeOf(Environment) and oppositeEnd.oclIsTypeOf(Environment)) or
(self.oclIsTypeOf(Vehicle) and oppositeEnd.oclIsTypeOf(Vehicle)) or
(self.oclIsTypeOf(Goal) and oppositeEnd.oclIsTypeOf(Goal)) or
(self.oclIsTypeOf(Task) and oppositeEnd.oclIsTypeOf(Task)) or
(self.oclIsTypeOf(Evaluator) and oppositeEnd.oclIsTypeOf(Evaluator)) or
(self.oclIsTypeOf(Actuator) and oppositeEnd.oclIsTypeOf(Actuator)) or
(self.oclIsTypeOf(KComponent) and oppositeEnd.oclIsTypeOf(KComponent)) or
(self.oclIsTypeOf(PComponent) and oppositeEnd.oclIsTypeOf(PComponent)) or
(self.oclIsTypeOf(VComponent) and oppositeEnd.oclIsTypeOf(VComponent)) or
(self.oclIsTypeOf(EComponent) and oppositeEnd.oclIsTypeOf(EComponent)))
```

Figura 16: Restricciones OCL para las relaciones *EInherits*.

```
self <> oppositeEnd and self.oclIsTypeOf(GeneralRelationship) and
oppositeEnd.oclIsTypeOf(GeneralRelationship)
```

Figura 17: Restricción OCL para las relaciones *RInherits*.

4.3 Generador de código

El generador de código es la herramienta de desarrollo encargada de llevar a cabo las distintas funcionalidades relacionadas con la modificación del código fuente original generado por Eclipse, y la especialización de ese código conforme a los requisitos que presente la plataforma de simulación de destino. La mayoría de las operaciones están parcialmente automatizadas a través de asistentes con el fin de proporcionar ayuda y orientar a los usuarios durante el proceso. Esto permite que los expertos en tráfico jueguen un rol principal, mientras que los programadores simplemente colaboran en las distintas fases donde la inserción o modificación del código fuente es indispensable. Un editor de texto, un editor gráfico y un compilador están integrados en la herramienta con el fin de apoyar estas funcionalidades.

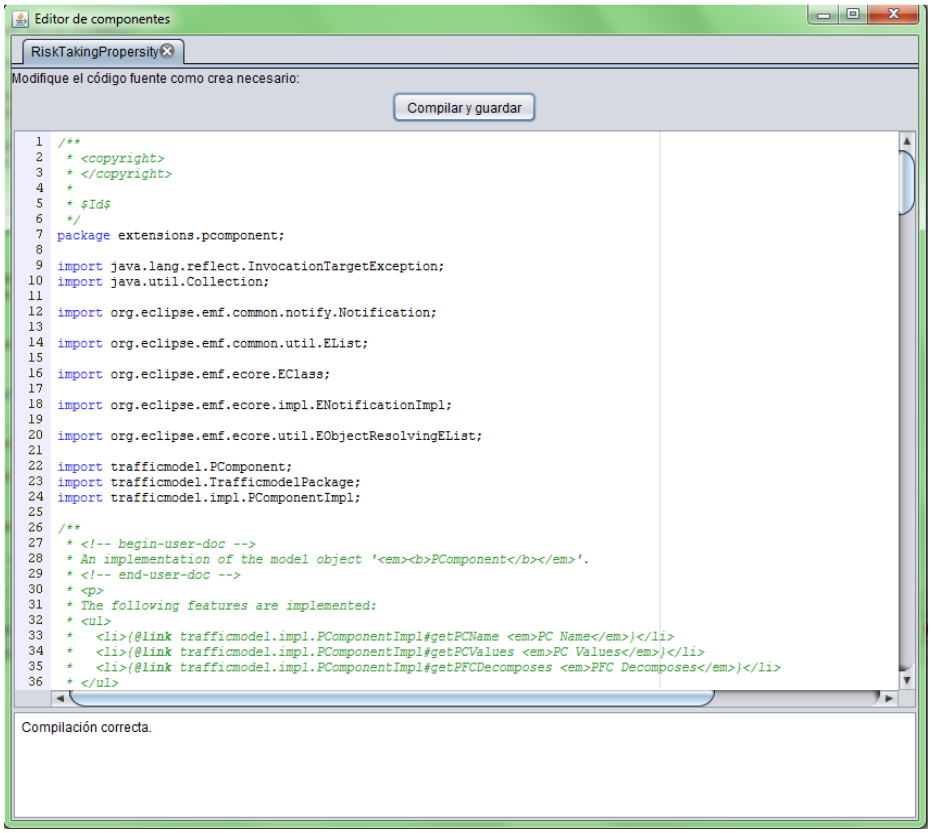

Figura 18: Editor de texto del generador de código.

El editor de texto trabaja con código fuente escrito en Java (ver Figura 18). Incluye las funcionalidades básicas (como puede ser el resaltado de palabras clave o la realización de búsquedas de texto especifico). Permite a los usuarios disponer de un entorno amigable que posibilita la compilación y el almacenamiento de las

modificaciones introducidas en el código fuente, junto a los comentarios destinados a la generación de documentación.

El editor gráfico (ver Figura 19) permite diseñar especificaciones de modelos basadas en el clúster *Interactivo* del metamodelo de manera análoga a como lo hace el plug-in de Eclipse (ver Sección 4.2). Se utiliza como soporte en la funcionalidad de integración de especificaciones.

El compilador integrado en la herramienta está formado por una instancia nueva del compilador de Java. Esta instancia se configura para que permita la creación, inclusión y compilación de nuevas clases en tiempo de ejecución utilizando reflexión.

Respecto al funcionamiento de la herramienta, ésta toma principalmente como entrada un fichero XMI que describe los elementos de la especificación de la simulación. Es el fichero creado por el editor gráfico (ver Sección 4.2) para almacenar el diseño del modelo. Una vez cargado se presenta una interfaz gráfica que permite la visualización de la información capturada en la especificación de modelo y una navegación intuitiva por los elementos del mismo (ver Figura 20). Seleccionando un elemento determinado se puede obtener información respecto a los métodos que tiene asociados, los elementos de los que se compone (ver Sección 3.2) las instancias de *GeneralRelationship* en las que actúa como origen, y los elementos del mismo tipo de los que hereda.

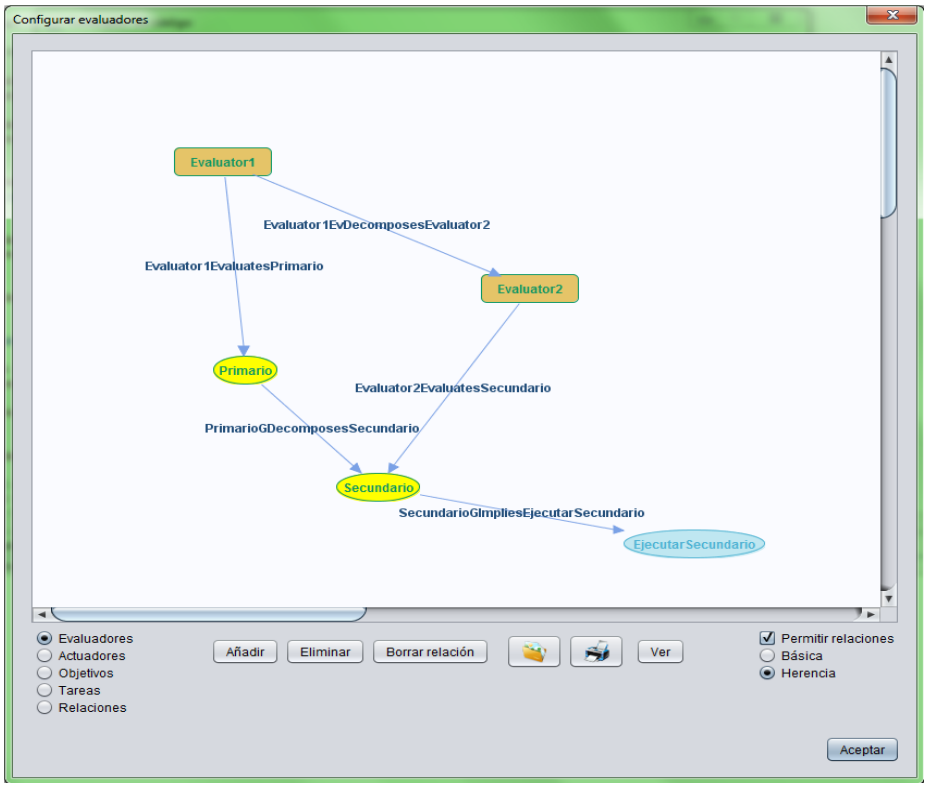

Figura 19: Editor gráfico embebido en el generador de código.

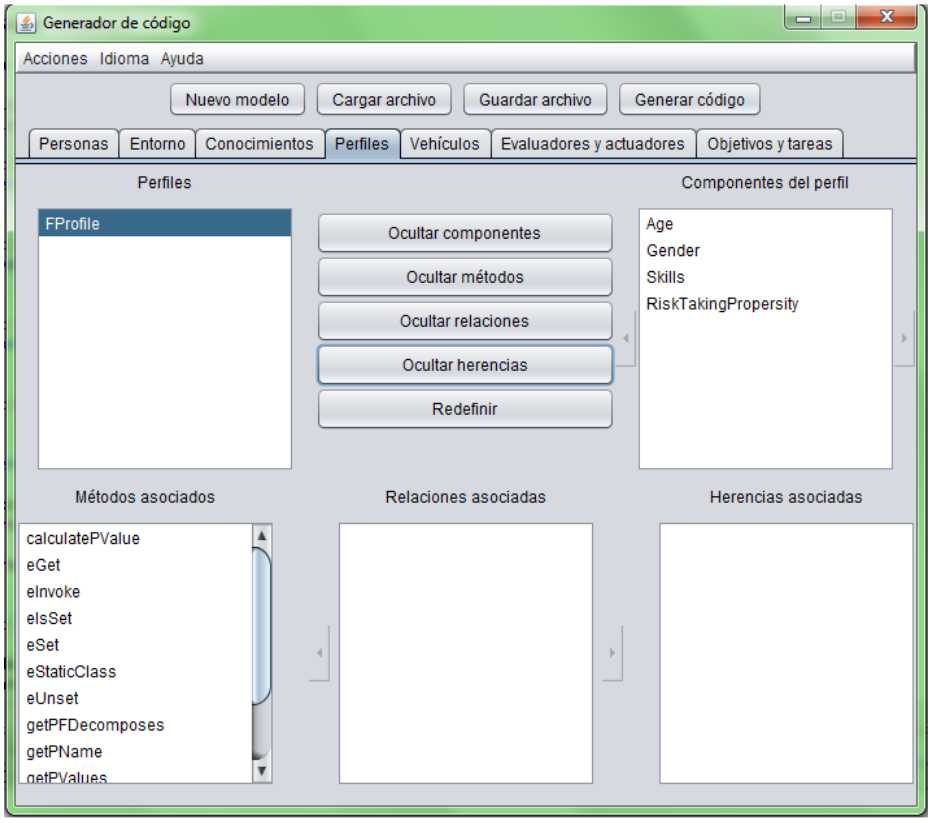

Figura 20: Interfaz gráfica del generador de código.

En cuanto a las funcionalidades relacionadas con la generación de código, la herramienta permite la inyección de código fuente, el diseño y el almacenamiento de modelos incompletos basados en el clúster *Interactivo*, y la integración de clústeres.

La inyección de código fuente ofrece dos alternativas con el fin de modificar y compilar dinámicamente las plantillas de código generadas por el EMF por defecto. La primera de ellas consiste en la modificación sólo del cuerpo de los métodos de las clases. Ello se hace mediante la inclusión de nuevas instrucciones utilizando fragmentos de código adecuados para la plataforma de simulación de destino. La segunda es más compleja, siendo capaz de modificar una clase completamente. Ello permite añadir nuevos atributos y métodos. Ya que estas operaciones requieren algunos conocimientos de programación, la interfaz gráfica y los módulos integrados (en este caso el editor de texto y el compilador) ayudan a examinar el código de los elementos del modelo. Por ejemplo, la primera permite configurar mediante un asistente gráfico una nueva clase que extiende de otra ya creada, mientras que los segundos posibilitan la extracción de un determinado método del resto de la clase para realizar las modificaciones en el cuerpo del mismo de manera simplificada. Esto facilita estas tareas y produce un

entorno de desarrollo más intuitivo. También hay ayuda en línea y ejemplos para guiar a los usuarios en este punto.

El diseño y el almacenamiento de modelos basados en el clúster *Interactivo* (ver Sección 3.2.3) utiliza el editor gráfico integrado. Éste es similar al editor gráfico explicado anteriormente (ver Sección 4.2). Proporciona un lienzo y una paleta para crear los distintos elementos (instancias de *Goal*, *Task*, *Evaluator*, *Actuator* y *GeneralRelationship*) y una herramienta de validación. Después de eso, el clúster generado se puede utilizar en la herramienta con el fin de realizar otras tareas, como la inyección de código o almacenarlo para futuras utilizaciones.

La funcionalidad de integración de clústeres enlaza con la anterior. Consiste en la combinación de una especificación de modelo que sólo incluye elementos de los clústeres *Mental* y *Entorno* con un modelo del clúster *Interactivo* o viceversa. Un asistente gráfico permite examinar los elementos disponibles, y proporciona indicaciones de las posibles referencias entre ellos de acuerdo con el LMT. Además, se pueden añadir instancias de *GeneralRelationship*. Esta funcionalidad permite reutilizar modelos basados en el clúster *Interactivo* con diferentes modelos de los clústeres *Mental* y *Entorno*. Una vez que se completa la integración, el resto de las funcionalidades de la herramienta se puede utilizar considerando la nueva especificación integrada del modelo.

En cuanto a la adaptación de la plataforma, la herramienta de generación de código facilita la especialización del diseño. Para ello incorpora dos funcionalidades principales: la inserción dinámica en el compilador y la generación de nuevas clases.

La inserción dinámica se encarga de añadir librerías (y las dependencias de las mismas). Estas librerías proceden generalmente de la plataforma de destino. Una vez que se selecciona la librería, el proceso es administrado internamente por la herramienta, por lo que se consigue que sea transparente a los usuarios. Permite la inserción de elementos no relacionados con el LMT en el código fuente de una clase de la especificación de modelo.

La generación de nuevas clases es una funcionalidad que se lleva a cabo con la ayuda de un asistente. Las clases creadas pueden ser completamente nuevas, o por el contrario, heredar de las originales creadas por EMF o de las procedentes de las librerías externas. Estas clases son consideradas dinámicamente por el compilador proporcionado por la herramienta. Esta funcionalidad permite la instanciación y utilización de las mismas en el resto de clases del proyecto.

Cuando se completan la transformación de código y la adaptación de plataforma, se produce un archivo final. Este puede ser generado a través de dos tipos de enfoques, un plug-in específico o una nueva plataforma completa. En ambos casos, el proceso se apoya en asistentes para hacerlo más intuitivo.

El enfoque basado en un plug-in específico crea un archivo comprimido que empaqueta la especificación de modelo, las librerías necesarias, y las clases modificadas y generadas. El archivo puede ser cargado en la plataforma de destino.

El enfoque basado en una nueva plataforma integra la simulación (especificación de modelo y las clases modificadas y generadas) con toda la plataforma de simulación de destino y sus librerías dependientes si son necesarias. El resultado es un archivo ejecutable comprimido. La plataforma resultante es capaz de ejecutar simulaciones considerando la especificación de modelo insertado y configurando después las instancias a considerar de sus tipos.

También se pueden incluir archivos de configuración. Estos establecen los posibles valores iniciales de los elementos de la especificación de modelo.

4.4 Plataforma de simulación

La plataforma de simulación está implementada en Java. Permite llevar a cabo simulaciones de tráfico microscópicas reproduciendo características del entorno real. Considera los roles de tráfico *conductor* y *peatón*, utilizando agentes inteligentes para la implementación de los individuos. Para ello utiliza una serie de instrumentos:

- La plataforma de agentes A-Globe [82]. Está especializada en el desarrollo rápido de SMA. Es apropiada para realizar simulaciones realistas del entorno de tráfico ya que dispone de Sistemas de Información Geográfica (GIS, *Geographic Information System*) [12]. Respecto a la arquitectura de la plataforma (ver Figura 21), ésta se compone de cinco elementos fundamentales:
  - *Plataforma de agentes*: proporciona los componentes básicos para que los agentes se puedan ejecutar.
  - *Contenedores de agentes*: es la entidad estructural de la plataforma. Asegura funciones básicas, infraestructura de comunicación y almacenamiento de agentes.
  - *Servicios*: proporcionan funciones comunes a los agentes de un mismo contenedor.
  - *Agente Simulador del Entorno* (ES, *Environment Simulator*): simula el entorno real y controla la visibilidad entre los contenedores de agentes.
  - *Agentes*: representan las entidades funcionales básicas en un determinado escenario.

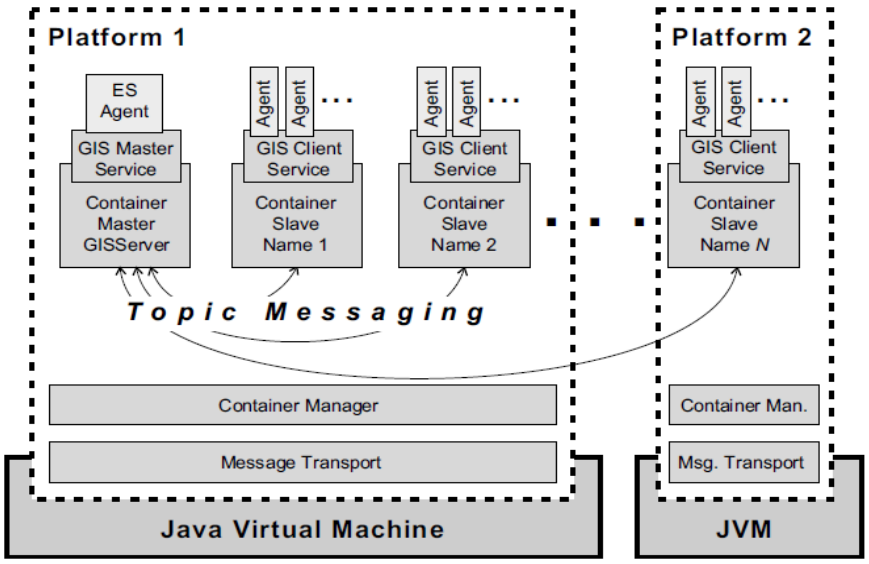

Figura 21: Estructura de la plataforma de agentes A-Globe [82].

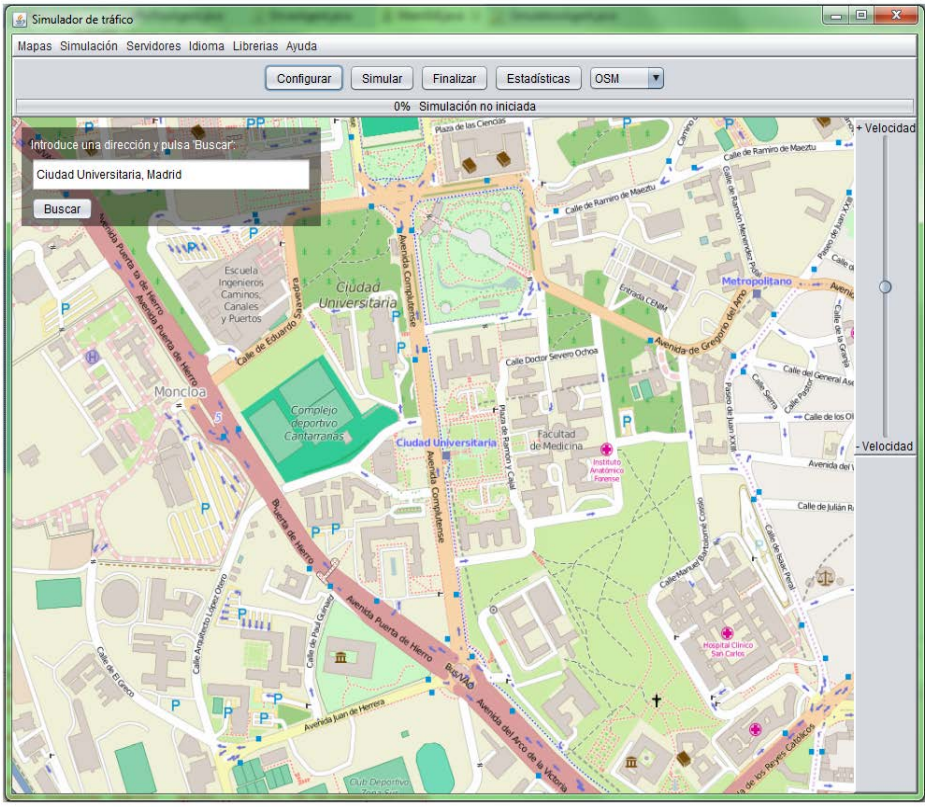

Figura 22: Interfaz gráfica de la plataforma de simulación.

- El *Kit de Desarrollo de Software* (SDK, *Software Development Kit*) de *Esri* [51]. Es utilizado para la integración de mapas en aplicaciones. Permite la representación gráfica de los entornos y dispone de geolocalización para mostrar direcciones precisas a través de la interfaz gráfica del mapa (ver Figura 22). Esta interfaz permite varios tipos de vistas (satélite, híbrida o mapa), y dispone de un zoom para acercar o alejar la vista sobre el entorno. También se encarga de mostrar los individuos en el entorno y los distintos movimientos e interacciones que llevan a cabo.
- Los servicios web de OpenStreetMap [28]. Están dedicados a la obtención de los elementos del entorno y su información (tipos de vías, infraestructuras o cualquier otro aspecto de importancia para la simulación). Proporcionan los datos mediante ficheros XML que contienen la información geográfica y de interés del entorno, además de las coordenadas de los distintos elementos.
- Las librerías de navegación y de datos de OpenStreetMap [28]. Las primeras son utilizadas para la obtención y generación de las rutas de los individuos conforme a las coordenadas de un determinado fragmento del entorno. Estas librerías son capaces de proporcionar distintas rutas para peatones o vehículos según el tipo de algoritmo aplicado. Además, ofrecen la posibilidad de configurar la

obtención de las rutas según una serie de métricas (por ejemplo, la ruta más corta o la que menos tiempo estimado conlleve según la velocidad máxima permitida en las vías). Las segundas se emplean para generar las estructuras de datos que almacenan la información obtenida por los servicios web. Estos datos se mapean en la plataforma de test a una matriz bidimensional que construye y ubica los elementos en el modelo interno. Este modelo permite a los agentes conocer las características del entorno e interactuar entre ellos.

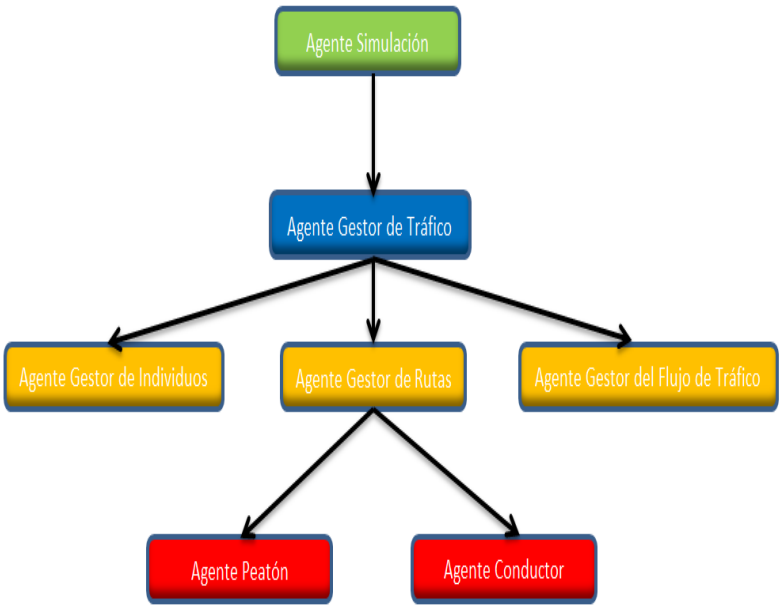

Figura 23: Estructura jerárquica de los agentes de la plataforma.

En cuanto a la estructura de agentes de la plataforma, estos se organizan de manera jerárquica (ver Figura 23), dentro de un mismo contenedor de la plataforma A-Globe [82]. El agente principal es el agente *Simulación*, el cual se encarga de la creación y actualización de la interfaz gráfica y de la comunicación con el modelo de la aplicación. El agente *Gestor de Tráfico* es creado por el agente *Simulación* y centraliza el proceso. Recibe información del entorno del agente *Simulación* y está encargado de crear y comunicarse con los agentes *Gestor de Individuos*, *Gestor de Rutas* y *Gestor del Flujo de Tráfico*. El agente *Gestor de Individuos* controla los individuos utilizados durante la simulación. Este agente se ocupa de crear agentes *Peatón* y *Conductor*, con sus distintas configuraciones individuales, o de grupo, según la cantidad de individuos deseada por el usuario. El agente *Gestor de Rutas* obtiene las posibles rutas transitables del entorno y las recalcula conforme al modelo de simulación. Estas rutas se envían a cada uno de los agentes *Peatón* y *Conductor* mediante el agente *Gestor de Tráfico*. El agente *Gestor del Flujo de Tráfico* se ocupa del funcionamiento y sincronización de los semáforos que aparecen en el entorno seleccionado por el usuario. Para ello implementa heurísticas basadas en el estudio del flujo de vehículos (es decir, el tránsito vehicular según [10]). Éste permite generar distintas configuraciones de la sincronización de los

semáforos para simular fenómenos tales como *la ola verde* [76]. Respecto a los agentes *Peatón* y *Conductor*, estos presentan un objetivo principal simple consistente en completar su ruta lo más rápido posible evitando atropellos en el caso de los primeros y colisiones en el caso de los segundos.

También se ha desarrollado una aplicación servidor para su uso desde la plataforma de test. La plataforma puede generar por sí misma las rutas que utilizan los individuos, pero delegar este trabajo en el servidor permite reducir su carga de trabajo. De esta manera, la plataforma de test puede centrarse en las operaciones requeridas por el estado de la simulación junto a su representación gráfica. Ambas aplicaciones se conectan mediante protocolo TCP/IP (*Transmission Control Protocol / Internet Protocol)* posibilitando a la plataforma de test compartir la información del entorno con el servidor, y al servidor proporcionar las rutas obtenidas a la plataforma. Utilizan programación basada en hebras, lo cual les permite disponer de procesos independientes para cada uno de los cálculos de ruta. Además, gracias a esto ambas pueden ser ejecutadas de manera óptima en clústeres de cálculo científico que realicen las distintas operaciones necesarias en múltiples procesadores simultáneamente.

4.5 Conclusiones

Disponer de las herramientas de desarrollo necesarias para una aproximación de ISDM puede tener un coste elevado. Existen plataformas de desarrollo (por ejemplo, las de Eclipse) capaces de proporcionar herramientas básicas con un esfuerzo relativamente bajo, pero los usuarios necesitan herramientas adaptadas a sus necesidades para obtener todo el provecho de estas propuestas. Por este motivo, este trabajo ha optado por desarrollar herramientas adaptadas al contexto de las simulaciones de tráfico, haciendo uso siempre que ha sido posible de infraestructuras ya existentes.

El editor gráfico permite modelar y reutilizar especificaciones de modelos de una manera gráfica e intuitiva. Ello facilita trabajar en el modelado con el LMT de nuevas teorías de tráfico y simulaciones. Entre las desventajas de su implementación actual están los problemas para adaptarlo automáticamente a cambios en los ficheros encargados de generarlo o el metamodelo, ya que obligan a revisar todos los elementos y se pierden los cambios realizados manualmente con anterioridad.

Por su parte, el generador de código facilita la transición al código desde los modelos. Dispone de una interfaz gráfica que permite la navegación de las especificaciones, y múltiples asistentes gráficos para soportar sus distintas funcionalidades. Fomenta la reutilización de especificaciones de modelos (tanto completas como parciales) y del código generado o modificado. Gracias a esta herramienta, el código especializado para una plataforma de destino determinada se puede implementar una sola vez, y reutilizar y mejorar en proyectos sucesivos.

La plataforma de test permite probar toda la infraestructura propuesta de ISDM y validarla. Para ello puede ser integrada como librería externa en el generador de código (empaquetándola en un archivo comprimido), o ser configurada mediante código fuente para que considere un plug-in producido por el generador de código. En el primer caso el objetivo consiste en generar una nueva plataforma que integre una especificación de

modelo determinada, mientras que en el segundo se modifica la plataforma original para adaptarla a la especificación de modelo producida. Esto permite que los distintos comportamientos basados en teorías de tráfico puedan ser simulados y verificados en un entorno controlado por los desarrolladores. Además, facilita comprobar el correcto funcionamiento del resto de las herramientas del marco de desarrollo.

Estas herramientas se encuentran en continua evolución y optimización para abordar nuevas necesidades que puedan surgir. Por ejemplo, se pueden añadir funcionalidades nuevas al generador de código para considerar nuevos elementos del metamodelo. Aquí un punto a mejorar es facilitar los cambios en nuestras herramientas superando la rigidez mostrada por EMF [85] y principalmente por GEF [73] para las modificaciones. En la actualidad, realizar extensiones o cambios en el metamodelo obliga a elaborar nuevamente de manera completa todos los elementos usados en el desarrollo del editor gráfico. Estos son los incluidos en el archivo con extensión *gmfmap*. Los proyectos de Eclipse no soportan su modificación parcial para contemplar cambios en el metamodelo.

Finalmente, indicar que todas estas herramientas y sus múltiples funcionalidades apoyan el enfoque de ISDM propuesto para las simulaciones de tráfico. Facilitan el proceso introduciendo asistentes que guían a los usuarios en las tareas más complejas, y permiten la integración de múltiples objetos en un único proyecto (por ejemplo, las especificaciones de modelos o fragmentos de código). Además, fomentan la reutilización y el desarrollo incremental, reduciendo la codificación manual en múltiples fases del proceso.

Capítulo 5. Proceso de desarrollo

En este capítulo se aborda el proceso de desarrollo para llevar a cabo una simulación usando la infraestructura de ISDM vista en los capítulos anteriores. El punto de partida son las teorías de tráfico ya existentes y la plataforma de simulación de tráfico que van a ser utilizadas en la simulación. El proceso especifica la forma de elaborar a partir de esos elementos los modelos de la simulación, y cómo obtener de ellos el código fuente.

5.1 Introducción

Este capítulo de tesis describe el proceso de desarrollo encargado de simular teorías de tráfico existentes en una plataforma destino mediante la infraestructura de ISDM. Se describe utilizando SPEM [56]. El proceso se introduce partiendo de sus fases principales, y pasando después a describir sus fases internas, las tareas que realizan, las herramientas que usan, y los roles involucrados.

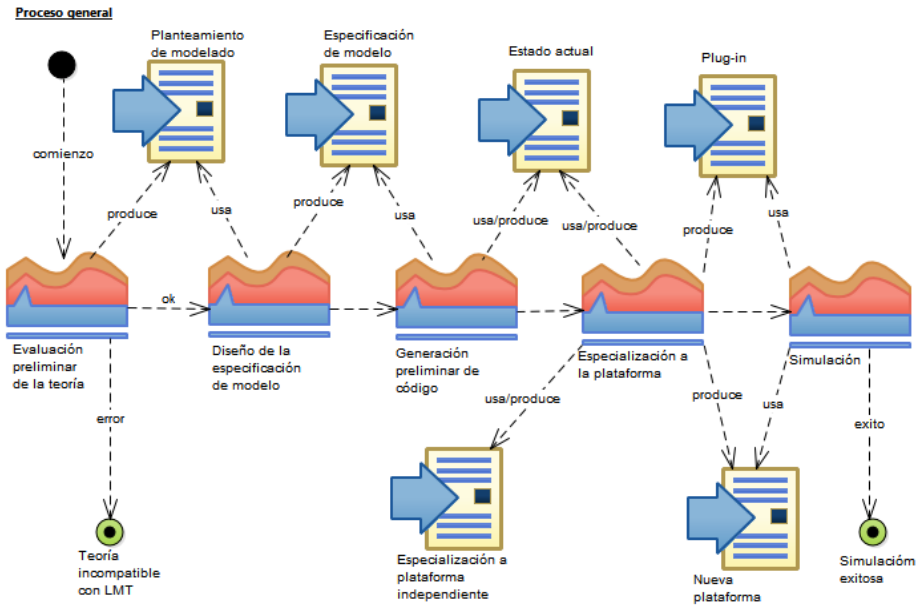

Figura 24: Proceso general de desarrollo.

El proceso de desarrollo mediante el cual llevar a cabo simulaciones de tráfico debe cubrir desde el momento en que los expertos estudian el problema y eligen las teorías a modelar, hasta el instante en que la simulación se ejecuta en la plataforma destino. En este proceso pueden identificarse cinco fases principales (ver Figura 24): *Evaluación preliminar de la teoría*, *Diseño de la especificación de modelo*, *Generación preliminar de código*, *Especialización a la plataforma*, y *Simulación*. Aunque la representación del proceso en el diagrama muestra un flujo de trabajo lineal con el propósito de ganar claridad, el proceso debe entenderse como incremental e iterativo. Esto implica que los usuarios pueden volver desde una fase a cualquiera de las fases anteriores y continuar desde ella.

En cuanto a los roles de los usuarios implicados en el proceso, hay dos principales: el *experto en tráfico* y el *programador*. El primero lleva a cabo la mayor parte de las tareas, evaluando la teoría de tráfico seleccionada, diseñando el modelo correspondiente y validando la simulación. El segundo colabora en las fases *Generación preliminar de código* y *Especialización a la plataforma*. Se ocupa de codificar las operaciones (por ejemplo, modificaciones en el cuerpo de los métodos predefinidos o programación de fórmulas matemáticas), e introducir nuevos elementos (por ejemplo, clases específicas o métodos para llevar a cabo la especialización de la especificación de modelo a la plataforma destino).

La fase principal *Evaluación preliminar de la teoría* es la primera del proceso. En ella los expertos evalúan la teoría de tráfico seleccionada e intentan realizar un posible planteamiento de su futura especificación con el LMT. Si se comprueba que este planteamiento es aceptable, entonces se continúa con el proceso en la siguiente fase. Por el contrario, si no se consigue plantear una especificación apropiada de la teoría, entonces se descarta. Alternativamente se puede considerar una revisión del LMT (y consiguientemente del resto de la infraestructura) para adaptarla a los requisitos presentados por dicha teoría.

En la fase principal *Diseño de la especificación de modelo* se procede a diseñar la especificación del modelo según el planteamiento obtenido en la fase anterior. En esta fase los expertos en tráfico realizan las modificaciones de manera incremental, añadiendo nuevos elementos y comprobando que cumplen las restricciones impuestas por el LMT. Una vez que la especificación está completa y validada, se puede continuar con la siguiente fase.

La fase principal *Generación preliminar de código* está encargada de generar el código fuente utilizando la especificación de modelo anterior. Esta fase consiste en la introducción de fragmentos de código específico en las clases extendidas con el generador de código. Estas clases se generan a partir de las plantillas originales producidas por EMF [85]. El generador de código soporta con asistentes tanto las operaciones de creación de métodos y atributos en las clases como la inserción de código en el cuerpo de los primeros. También se encarga de compilar las modificaciones introducidas y almacenar el estado actual del proyecto. Por ejemplo, este código insertado se emplea para codificar algunas fórmulas matemáticas o cuantificar la influencia de las interacciones entre individuos sobre un atributo.

En la fase principal *Especialización a la plataforma* se modifica el código fuente para adaptarlo según los requisitos de la plataforma destino. Para ello se introducen clases que empaqueten las especificaciones de modelos y se desarrolla la toma de decisiones de los individuos. Esta toma de decisiones se lleva a cabo conforme a los

aspectos considerados en las especificaciones y a los elementos proporcionados por la plataforma.

Esta fase presenta una fase interna opcional que permite reutilizar especificaciones del clúster *Interactivo* para completar especificaciones que no incluyan este clúster. Esto es debido a que el código del clúster *Interactivo* suele contener elementos específicos de la plataforma para la percepción y actuación, y a la existencia de teorías de tráfico especializadas sólo en las características de los individuos o en su toma de decisiones. La siguiente fase interna se encarga de realizar especializaciones de plataforma mediante la adaptación de una especificación de modelo completa (incluyendo los tres clústeres). Además se permite la generación de un archivo de configuración de los parámetros de la simulación. Esta tarea es soportada por asistentes gráficos que guían a los usuarios en su implementación.

El resultado final de esta fase puede ser dos tipos de archivos comprimidos. El primero es un plug-in para la plataforma destino que permite su utilización en ésta como librería. De esta manera, la plataforma puede ser modificada mediante código para que considere los aspectos de la especificación de modelo. Esto permite realizar plataformas de tráfico especializadas en el LMT propuesto en esta tesis. El segundo realiza una integración de la especificación de modelo y la plataforma de destino, generando una nueva plataforma de tráfico ya modificada que engloba los aspectos de la especificación. En ambos casos se facilita la incorporación del correspondiente archivo de configuración. Este archivo puede ser modificado para generar variaciones en el comportamiento de los individuos considerados en la simulación de tráfico.

Finalmente, la frase principal *Simulación* conforma la simulación modificando los parámetros presentados en el archivo de configuración y se encarga de la correcta ejecución de la misma. Si el archivo producido en la anterior fase es un plug-in, en esta fase debe ser añadido como librería en la plataforma de destino. Posteriormente, la plataforma ha de ser adaptada para que las simulaciones que se lleven a cabo en ella consideren las nuevas funcionalidades proporcionadas por dicha librería.

El resto del capítulo describe estas fases principales con más detalle en la Sección 5.2. La Sección 5.3 presenta las conclusiones obtenidas, señalando los puntos críticos del proceso.

5.2 Fases del proceso de desarrollo

Las cinco fases principales del proceso se descomponen en varias fases internamente. Al igual que las fases principales, estas fases son mostradas de manera lineal por simplicidad, pero también se conciben como iterativas e incrementales. Ello permite a los usuarios volver a cualquiera de las etapas anteriores dentro de la fase principal correspondiente en caso de ser necesario. Nótese que algunas de las mismas tienen bucles de manera explícita. Estos bucles subrayan la necesidad de que se cumplan ciertas condiciones para continuar con la siguiente fase. Por ejemplo, las nuevas modificaciones que se introduzcan en la especificación de modelo deben ser validadas para poder continuar.

El resto de la sección se organiza como sigue. La Sección 5.2.1 describe la fase de *Evaluación preliminar de la teoría*, la 5.2.2 la fase de *Diseño de la especificación de*

modelo, la 5.2.3 la fase de *Generación preliminar de código*, la 5.2.4 presenta en detalle la fase de *Especialización a la plataform*a profundizando en los dos tipos de archivos resultantes que produce, y la 5.2.5 analiza la fase de *Simulación* y sus dos posibles opciones basadas en los archivos de la fase anterior. Un ejemplo se introduce en cada sección para mostrar el funcionamiento del proceso en cada una de estas fases.

5.2.1 Evaluación preliminar de la teoría

El proceso de desarrollo comienza con la fase de *Evaluación preliminar de la teoría*. Esta primera fase principal consta de cuatro fases internas (ver Figura 25): *Estudio del problema a modelar*, *Selección de la base teórica*, *Estudio de la viabilidad* y *Aceptación de la teoría*. En todas ellas el rol de usuario utilizado es del *experto en tráfico*, ya que es el conocedor principal de la literatura del dominio.

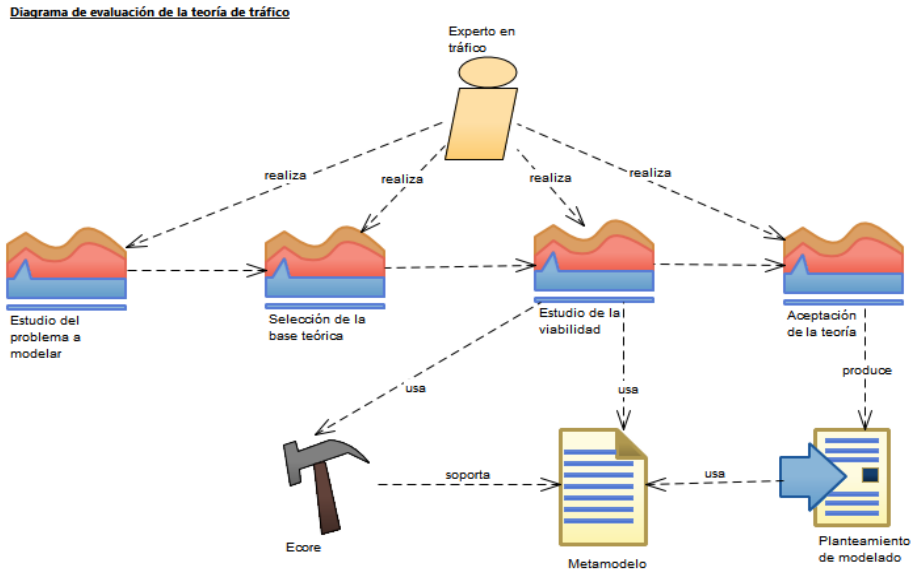

Figura 25: Proceso de evaluación de la teoría de tráfico.

En la primera fase interna los expertos evalúan el trabajo relacionado con el tráfico que se desea modelar. Para ello se realiza un estudio preliminar para comprobar que es compatible con el LMT, es decir, que puede ser modelado con él. Se empieza considerando el tipo de problema que aborda y si está enfocado en los individuos que participan en el tráfico. A continuación, se debe comprobar si considera el comportamiento de estos individuos, la toma de decisiones de los mismos o ambos. En caso afirmativo, se anotan los roles de los individuos implicados (por ejemplo, conductores o peatones) y qué tipo de interacciones se producen entre ellos (por ejemplo, si son interacciones entre individuos con el mismo rol o si por el contrario se consideran interacciones entre múltiples tipos de individuos). Además se debe observar que el trabajo seleccionado especifique un conjunto de características propias de los individuos o realice algún tipo clasificación de las decisiones de los mismos. En caso de que estas consideraciones hayan sido satisfechas se continúa hacia la siguiente fase

interna, ya que se considera que el trabajo puede ser modelado con el LMT. Por el contrario, si se comprueba que el trabajo evaluado presenta deficiencias o no encaja con lo esperado, debe de ser rechazado.

En la segunda fase interna los expertos de tráfico extraen los elementos teóricos del trabajo que se pueden hacer corresponder con los conceptos contenedores del LMT (ej. *Person*, *Profile*, *Environment* o *Vehicle*). Por ejemplo, estos trabajos pueden describir un entorno especifico (una ciudad o una intersección determinada), medios de transporte (vehículos o a pie) o una estructura de decisión sobre un conjunto de acciones. Si es posible establecer estas correspondencias, entonces se avanza hacia la siguiente fase interna del proceso. Por el contrario, si estas teorías están muy alejadas de los requisitos del LMT, entonces el trabajo es rechazado.

En la tercera fase interna se estima el nivel de ajuste de la teoría o teorías extraídas con el LMT realizando comparaciones entre sus conceptos, relaciones y propiedades. Este proceso comenzó con una primera aproximación centrada en los conceptos en la fase anterior. En esta fase se completa la evaluación comprobando que los distintos aspectos que los conforman puedan ser clasificados mediante alguna de las metaclases *XComponent* de los clústeres *Mental* y *Entorno* del metamodelo (ver Sección 3.2). Si se valora que algún elemento no puede ser representado por estas metaclases, se procede a comprobar su posible pertenencia a alguna de las metaclases que conforman el clúster *Interactivo*. Si no es así, la aproximación se considera no representable mediante la versión actual del LMT y es rechazada. Por el contrario, si todos los aspectos de las teorías pueden ser incluidos en alguno de los clústeres del metamodelo, entonces la evaluación es satisfactoria y se pasa a la siguiente fase interna.

En la cuarta fase interna se lleva a cabo el *planteamiento de modelado*. Gracias al filtro realizado en las fases internas anteriores, los expertos en tráfico pueden plasmar formalmente cómo se corresponden los aspectos de las teorías al LMT. En esta clasificación se indica la metaclase del LMT más apropiada para representar cada uno de los aspectos considerados por las teorías de tráfico. Si este planteamiento se completa satisfactoriamente, las teorías de tráfico extraídas del trabajo seleccionado son aceptadas. A continuación, el proceso avanza hacia la siguiente fase principal.

5.2.1.1 Ejemplo de evaluación preliminar de la teoría

Como ejemplo del proceso de desarrollo en esta primera fase principal se decide estudiar la posibilidad de adaptación al LMT de un trabajo existente en la literatura del dominio [68]. Este trabajo se evalúa en cada una de las fases internas. Una vez completadas, la teoría de tráfico obtenida será considerada compatible con el LMT y se habrá llevado a cabo de manera satisfactoria un planteamiento de modelado.

En la fase interna *Estudio del problema a modelar* se lleva a cabo un análisis sobre el trabajo seleccionado. Un primer estudio (ver Tabla 1) muestra un enfoque centrado en identificar los factores de riesgo y propensión a sufrir accidentes por parte de los conductores, y como estos factores afectan al comportamiento de los mismos a nivel microscópico. Se observa que sólo se tiene en cuenta el rol conductor, y que las interacciones que se detallan son homogéneas, es decir, únicamente entre individuos del mismo rol. Además, se introduce una clasificación de factores que pueden afectar al comportamiento de este tipo de individuos. Por ello, se considera que el trabajo satisface los requisitos de esta fase y que es provisionalmente compatible con el LMT.

Individual differences	**Situational factors**	
	Individual	*Environmental*
Age	Impairment	Time of day
Gender	Hurry/ Distraction	Day of week
Risk-taking propensity	Trip purpose	Weather
Skills	Length of drive	Road condition
Vehicle size		
Vehicle performance characteristics		

Tabla 1: Factores considerados por la teoría de tráfico de ejemplo.

La fase interna *Selección de la base teórica* consiste en la extracción de los conceptos teóricos principales en el trabajo seleccionado. En este caso, esta teoría se constituye mediante una serie de factores implicados en el comportamiento de los conductores. Estos factores están organizados en grupos de manera jerárquica. Se observa que en la aproximación no se indica un entono específico.

La clasificación de factores planteada reúne algunas de las características deseables para que la teoría sea compatible con el LMT. Así, se pueden identificar aspectos relacionados con las características del entorno (por ejemplo, *Road condition* o *Weather*), y aspectos vinculados con las características de los vehículos (por ejemplo, *Vehicle size*). Esto muestra una diferenciación evidente entre los medios de transporte y el resto del entorno, por lo que se puede deducir que la teoría puede ser compatible con el modelo CVE implícito en el LMT. Tras este análisis, se decide seguir evaluando la teoría extraída.

En la fase interna *Estudio de la viabilidad* se comprueba si los factores presentes en la teoría extraída son compatibles con las metaclases proporcionadas por el LMT. En particular, se puede observar que algunos de estos factores están relacionados con el conocimiento actual de las personas (por ejemplo, *Trip purpose* o *Length of drive*), o con las características propias de los individuos (por ejemplo, *Age* o *Gender*). Además, en la fase anterior se ha constatado la presencia de factores vinculados con el entorno y con los medios de transporte. En conclusión, completado el estudio no se detecta la presencia de ningún aspecto que no pueda ser representado por alguna de las metaclases de los clústeres *Mental* (ver Sección 3.2.1) y *Entorno* (ver Sección 3.2.2). Por ello se decide continuar hacia la siguiente fase interna. En el estudio se observa que no aparecen factores relacionados con las metaclases del clúster *Interactivo* (ver Sección 3.2.3). Ese hecho indica que la teoría extraída está formada por un conjunto de elementos que no contempla la toma de decisiones de los conductores.

La fase interna *Aceptación de la teoría* tiene por objetivo llevar a cabo el planteamiento de modelado. Si este planteamiento es satisfactorio, entonces las posibles teorías extraídas son aceptadas y en consecuencia el trabajo seleccionado que las contiene se considera compatible con el LMT. En este caso, en la anterior fase principal se ha descartado la presencia de aspectos de la teoría extraída que puedan ser presentados por las metaclases que constituyen el clúster *Interactivo*. Esto facilita la construcción del planteamiento de modelado, ya que el número de metaclases a considerar es menor.

La construcción comienza con la identificación de todos los factores de la teoría que deban ser situados dentro del clúster *Entorno*. Así, los factores *Vehicle size* y *Vehicle performance characteristics* correspondientes al medio de transporte (en este caso vehículos) se representan mediante la metaclase *VComponent*. Por su parte, los factores del resto del entorno *Time of day*, *Day of week, Weather* y *Road condition* son considerados por la metaclase *EComponent*.

Profile		**Environment**	**Knowledge**	**Vehicle**
Differences	*Individual*			
Age	Impairment	Time of day	Trip purpose	Vehicle Size
Gender	Hurry/ Distraction	Day of week	Length of drive	Vehicle performance characteristics
Risk-taking propensity		Weather		
Skills		Road condition		

Tabla 2: Planteamiento de modelado de la teoría de ejemplo.

A continuación, se identifican los elementos del clúster *Mental*. Como factores relacionados con las características de los individuos se seleccionan como principales *Individual differences* e *Individual*. El primero se descompone en *Age, Gender, Risk-taking propensity*, y *Skills*, mientras que el segundo se descompone en *Impairment y Hurry/Distraction*. Esto es debido a que en la teoría extraída se muestra una clasificación similar, aunque originalmente ésta contiene algunos factores que han sido reubicados en otras metaclases (por ejemplo, los factores vinculados con la metaclase *VComponent*). La metaclase *PComponent* es la encargada de este tipo de factores. En cuanto a los factores relacionados con el conocimiento actual se identifican *Trip purpose* y *Length of drive*. Estos también han sido reubicados de su clasificación original. La metaclase que los representa es *KComponent*. Finalmente, se comprueba que no aparecen aspectos de la teoría que no hayan sido identificados y ubicados en alguna de las metaclases que proporciona el LMT.

Las correspondencias anteriores indican que el planteamiento de modelado es factible (ver Tabla 2), aunque queda abierto a modificaciones futuras. De esta manera la teoría extraída es considerada modelable con el LMT.

5.2.2 Diseño de la especificación de modelo

En la segunda fase principal del proceso se diseña la especificación de modelo a partir del planteamiento de modelado producido anteriormente. Esta fase está formada por tres fases internas (ver Figura 26): *Construcción de la especificación de modelo*, *Validación de la especificación de modelo* y *Finalización de la especificación de modelo*. En todas ellas los usuarios que intervienen son aquellos con el rol de *experto en tráfico*.

En la literatura del dominio aparecen frecuentemente aproximaciones que presentan una marcada base teórica de descripción de características de los individuos y el entorno, pero que obvian la toma de decisiones de los individuos (ver Sección 5.2.3.1). Por ello se permite que las especificaciones de modelos puedan ser parciales, utilizando simplemente los clústeres *Mental* y *Entorno* para albergar una teoría determinada. En fases posteriores del proceso de desarrollo, esta especificación puede ser completada mediante otra especificación de modelo que contenga instancias de las metaclases del clúster *Interactivo* (ver Sección 5.2.4). Estas últimas suelen estar relacionadas con teorías basadas en la interacción entre individuos. Además, esto fomenta la reutilización de las teorías extraídas de los trabajos del dominio, permitiendo generar especificaciones de modelos completas que integren múltiples teorías y que puedan ser ajustadas de manera específica a una plataforma de tráfico destino.

En la primera fase interna los expertos en tráfico deben crear una *especificación de modelo* según los elementos identificados en el planteamiento de modelado. El procedimiento habitual comienza con la creación de un nuevo proyecto después de haber ejecutado el editor gráfico (ver Sección 4.2). A este proyecto se han de añadir los dos archivos XMI con formatos *trafficmodel* y *trafficmodel_diagram* que son necesarios para generar especificaciones de modelos acordes al LMT. Una vez completado este paso, se procede a generar un esquema básico de la especificación de modelo. Éste consiste en una especificación que presenta sólo las instancias de las metaclases principales (por ejemplo, *Knowledge* o *Environment*) de cada clúster considerado. Así, se añaden gráficamente las instancias de las metaclases de los clústeres *Mental* (ver Sección 3.2.1) y *Entorno* (ver Sección 3.2.2) que sean necesarias para satisfacer los requisitos de la clasificación realizada en el planteamiento de modelado. Una vez integradas las instancias para estos dos clústeres, se añaden las instancias de las metaclases principales proporcionadas por el clúster *Interactivo* (ver Sección 3.2.3) que hayan sido identificadas en el planteamiento de modelado. A continuación, siguiendo las directrices proporcionadas por el planteamiento de modelado, se crean e integran en la especificación de modelo las instancias de las metaclases *XComponent* (ver Sección 3.2) que forman parte de los clústeres *Mental* y *Entorno*.

Indicar que en cualquier momento se puede avanzar hacia la siguiente fase interna para validar el estado actual de la especificación de modelo. Una vez realizada esta validación se puede volver a esta fase interna en caso de ser necesario introducir nuevas modificaciones o para reparar algún error.

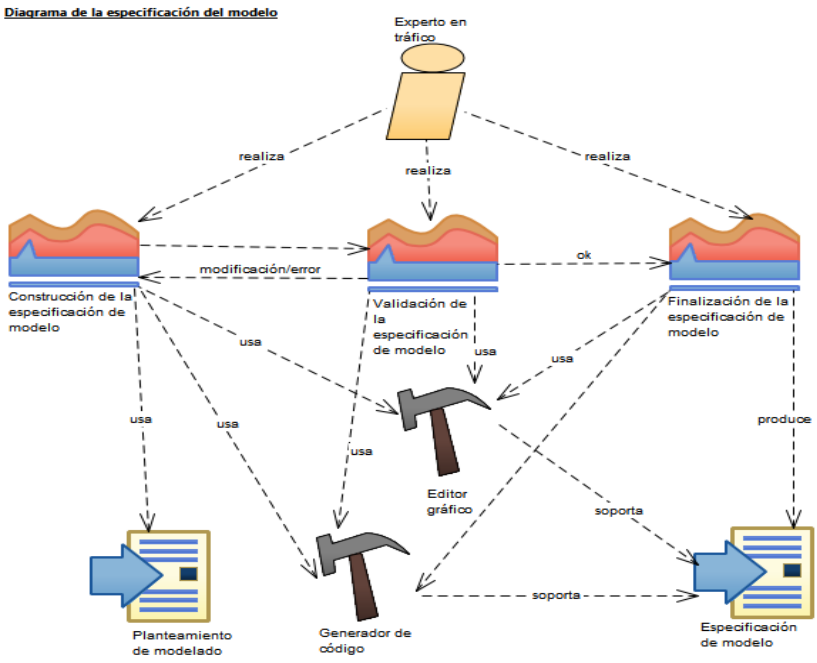

Figura 26: Proceso de obtención de la especificación de modelo.

La segunda fase interna se encarga de la validación del estado actual de la *especificación de modelo*. Esta validación es realizada mediante el editor gráfico, el cual identifica los posibles problemas que pudiesen presentarse. En ella se comprueba que la especificación de modelo sea compatible con el LMT. Para ello se comprueba que ésta cumple las restricciones de cardinalidad de las referencias entre elementos y satisface otras restricciones OCL introducidas (ver Sección 3.2). En el caso de que la validación sea satisfactoria, los expertos en tráfico pueden continuar con el proceso avanzando hacia la siguiente fase interna con el objetivo de finalizar el diseño de la especificación de modelo, o volver a la fase interna anterior para introducir nuevas modificaciones. Si la validación presenta errores, los expertos en tráfico deberán volver obligatoriamente a la fase interna anterior para llevar a cabo las modificaciones oportunas y poder continuar con el proceso.

En la tercera fase interna se realiza un estudio comparativo entre la *especificación de modelo*, el *planteamiento de modelado* y la teoría extraída del trabajo seleccionado. Este estudio tiene por objeto la detección de posibles errores de concordancia entre los nombres originales proporcionados por la teoría y los indicados en la especificación. Además, los expertos en tráfico deben revisar que todos los elementos identificados en el planteamiento de modelado estén representados apropiadamente en la especificación de modelo. En caso de que alguno de ellos no hubiese sido considerado, se debe volver a la primera fase interna para llevar a cabo su inserción en la especificación de modelo y continuar desde allí el proceso de desarrollo. Una vez que el estudio ha sido completado de manera satisfactoria, la especificación de modelo puede considerarse

finalizada (aunque quede abierta a nuevas modificaciones), pudiendo continuar hacia la siguiente fase principal.

El procedimiento alternativo consiste en plasmar el planteamiento de modelado en el editor gráfico incluido en el generador de código obteniendo una especificación de modelo que utilice sólo instancias del clúster *Interactivo*. Así, ésta debe incluir al menos una instancia *Evaluator* que represente la toma de decisiones y una instancia *Actuator* por cada tipo de individuo considerado (por ejemplo, conductores de perfil agresivo, conductores ebrios o peatones somnolientos). A continuación se agregan las instancias *Goal*. Éstas representan los deseos de los individuos indicados en el planteamiento de modelado y deben ser relacionadas con una instancia *Evaluator*. Por último se añaden las instancias *Task*. Cada instancia *Goal* debe conectarse con una de ellas y éstas sólo pueden relacionarse con una única instancia *Goal*. Tanto las instancias *Goal* como las *Task* pueden estar relacionadas también con otras de su mismo tipo. Las instancias *Task* deben relacionarse con la instancia *Actuator* correspondiente. Pueden añadirse instancias *GeneralRelationship* entre estas instancias de manera completa (es decir, con origen y destino determinado) o de manera parcial (es decir, dejando el origen o el destino sin indicar). No obstante, puede haber especificaciones que no necesiten alguna de las metaclases del clúster *Interactivo* para su funcionamiento, o que organicen las referencias entre las instancias de manera distinta, aunque esto no es recomendable. Si la herramienta lleva a cabo la validación de la especificación de modelo sin errores se puede asegurar que la especificación obtenida es compatible con el LMT. En caso contrario, la teoría de interacción extraída debe de ser revisada o el trabajo seleccionado rechazado. Finalmente, si existe compatibilidad con el LMT se debe realizar un estudio comparativo entre las teorías extraídas, el planteamiento de modelado y la especificación de manera similar al procedimiento habitual. Si este es satisfactorio entonces se puede continuar hacia la siguiente fase principal del proceso.

5.2.2.1 Ejemplo de diseño de la especificación de modelo

Continuando con el ejemplo (ver Sección 5.2.1.1), en esta fase principal se desarrolla la especificación de modelo partiendo del planteamiento de modelado. Debido a que el trabajo seleccionado sólo incluye elementos relacionados con los clústeres *Mental* y *Entorno*, se decide llevar a cabo esta fase siguiendo el procedimiento habitual.

En la fase interna *Construcción de la especificación de modelo* se comienza generando el proyecto y los archivos XMI necesarios. Una vez llevado a cabo esto, se procede a crear el esquema básico de la especificación de modelo. Para ello se añaden las instancias de las metaclases principales que disponen de elementos *XComponent* (ver Sección 3.2) en el planteamiento de modelado. Después se insertan las referencias entre estas instancias. Al no disponer de elementos en el planteamiento de modelado relacionados con el clúster *Interactivo* se obtiene un esquema básico con instancias procedentes solamente de los clústeres *Mental* y *Entorno* (ver Figura 27).

Una vez completado, es conveniente realizar una validación del estado actual de la especificación de modelo avanzando hacia la fase interna *Validación de la especificación de modelo*.

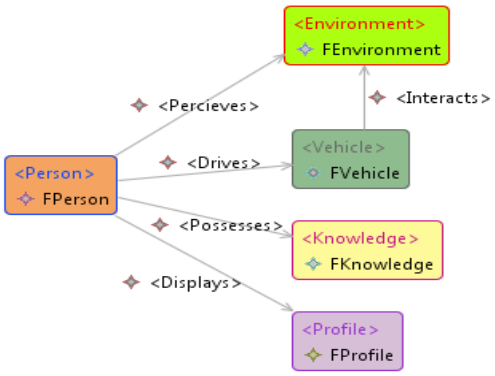

Figura 27: Ejemplo de esquema básico.

A continuación, se añaden el resto de instancias según la clasificación presentada en el planteamiento de modelado. Cada vez que se completa la composición de una instancia principal por medio de sus correspondientes instancias X*Component*, se avanza hacia la siguiente fase interna para validar el estado actual de la especificación. Aunque esta manera de proceder es opcional, es altamente recomendable, ya que favorece que la especificación de modelo se desarrolle de manera incremental, lo que reduce la aparición de errores y limita la complejidad de los que se presentan. Así, se procede a agregar e integrar en la especificación de modelo los elementos del planteamiento de modelado relacionados con el clúster *Entorno*.

Comenzando por los elementos de los vehículos, se crean dos instancias de la metaclase *VComponent*: *Size* y *PerformanceCharacteristics*. Estas instancias se integran con la instancia *FVehicle* mediante la referencia *VDecomposes*. Después se realiza una validación de la especificación. Si el resultado es el apropiado, se continúa con los elementos del planteamiento de modelado relacionados con el resto del entorno. Para ello se crean cuatro instancias *EComponent*: *TimeofDay*, *DayofWeek*, *Weather* y *RoadCondition*. Estas instancias se relacionan con la clase principal *FEnvironment* utilizando la referencia *EDecomposes*. Una vez validada de manera satisfactoria la especificación de modelo en este punto, se continúa con la creación de las instancias del clúster *Mental*. En el planteamiento de modelado se han creado dos clasificaciones diferenciadas de elementos relacionados con la metaclase principal *FProfile*. Por ello, dos instancias llamadas *IndividualReferences* e *Individual* de la metaclase *PComponent* son creadas para representar sus elementos principales. Estas instancias son integradas en la especificación mediante la referencia *PFDecomposes*. En el siguiente paso se introducen el resto de elementos de las dos clasificaciones. De esta manera, las instancias de *PComponent*: *Age*, *Gender*, *Skills* y *RiskTakingPropensity* son conectadas mediante la referencia *PFCDecomposes* a la instancia *IndividualReferences*, mientras que las instancias *Impairment* y *Hurry/Distraction* pasan a formar parte de la instancia *Individual* utilizando el mismo tipo de referencia. Una vez validada satisfactoriamente la especificación de modelo, se agregan dos instancias *KComponent*: *TripPurpose* y *LengthofDrive*. Estas instancias son anexadas a la instancia *FKnowledge* utilizando la referencia *KDecomposes*. Si la especificación generada se valida correctamente, entonces se puede avanzar hacia la última fase interna.

En la fase interna *Finalización de la especificación de modelo* se comprueba que todos los elementos del planteamiento de modelado aparecen en la especificación de modelo. Se evalúan las referencias establecidas entre instancias de manera que cumplan la clasificación establecida originalmente en el planteamiento. Además, se verifica que los nombres de las instancias coincidan con los considerados en la teoría extraída del trabajo del dominio [68] y con los del planteamiento de modelado. Indicar que en este caso, algunos nombres han sido modificados por simplicidad (por ejemplo, los elementos relacionados con los medios de transporte *Vehicle size* y *Vehicle performance characteristics* son representados por las instancias *Size* y *PerformanceCharacteristics*). Una vez completado este estudio comparativo se puede dar por finalizada la especificación de modelo (ver Figura 28).

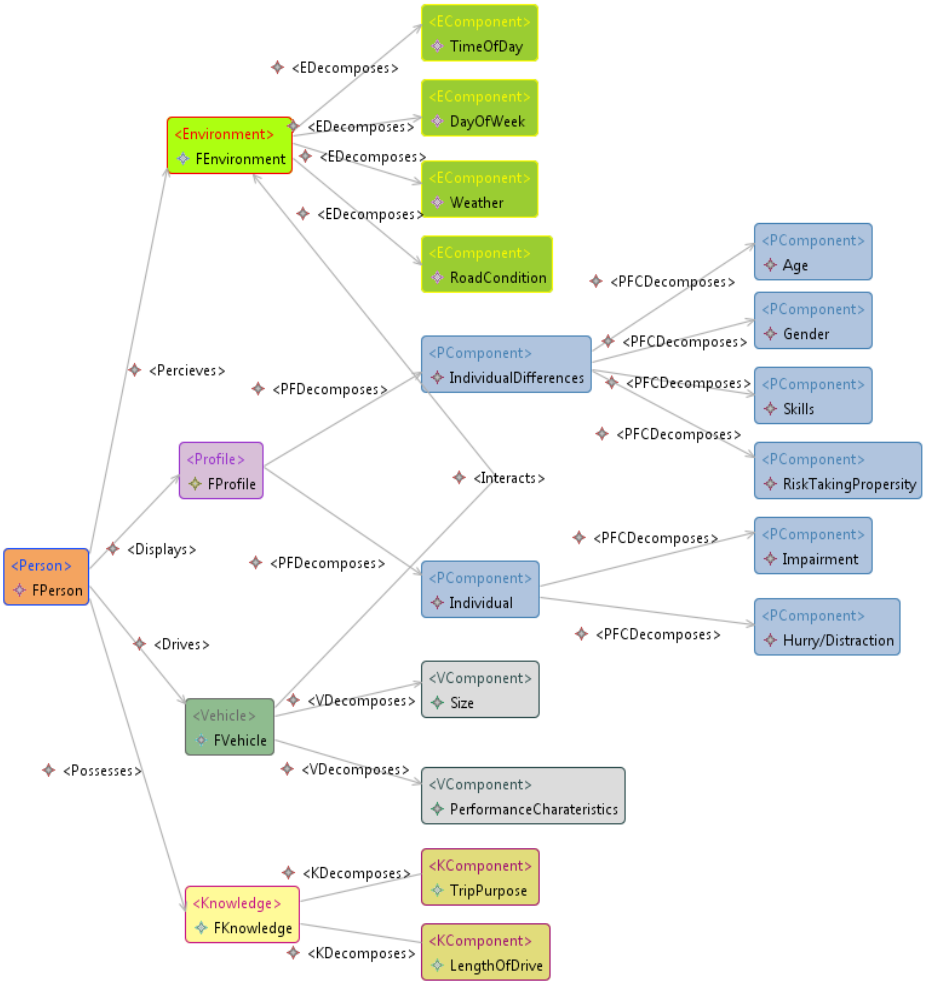

Figura 28: Especificación de modelo asociada a la teoría de tráfico.

5.2.3 Generación preliminar de código

La tercera fase principal del proceso de desarrollo engloba la transformación a código fuente de la especificación de modelo y la codificación preliminar de las operaciones realizadas por las instancias de las metaclases de los clústeres. Esta fase se lleva a cabo por usuarios con roles *experto en tráfico* y *programador*. Se descompone en dos fases internas (ver Figura 29): *Diseño de las interacciones* y *Codificación de las interacciones*. Estas fases presentan un procedimiento para las especificaciones parciales con instancias procedentes de los clústeres *Mental* y *Entorno*, y otro procedimiento diferente para las especificaciones de modelos parciales con únicamente un clúster *Interactivo*. Para especificaciones completas (es decir, con instancias relacionadas con los tres clústeres) se deben llevar a cabo ambos procedimientos. En este caso se recomienda comenzar por las modificaciones de código de las instancias de los clústeres *Mental* y *Entorno* para finalizar con los cambios en los métodos de las instancias del clúster *Interactivo*. Esto permite una visión más precisa sobre la influencia del estado de personas y entorno sobre la toma de decisiones.

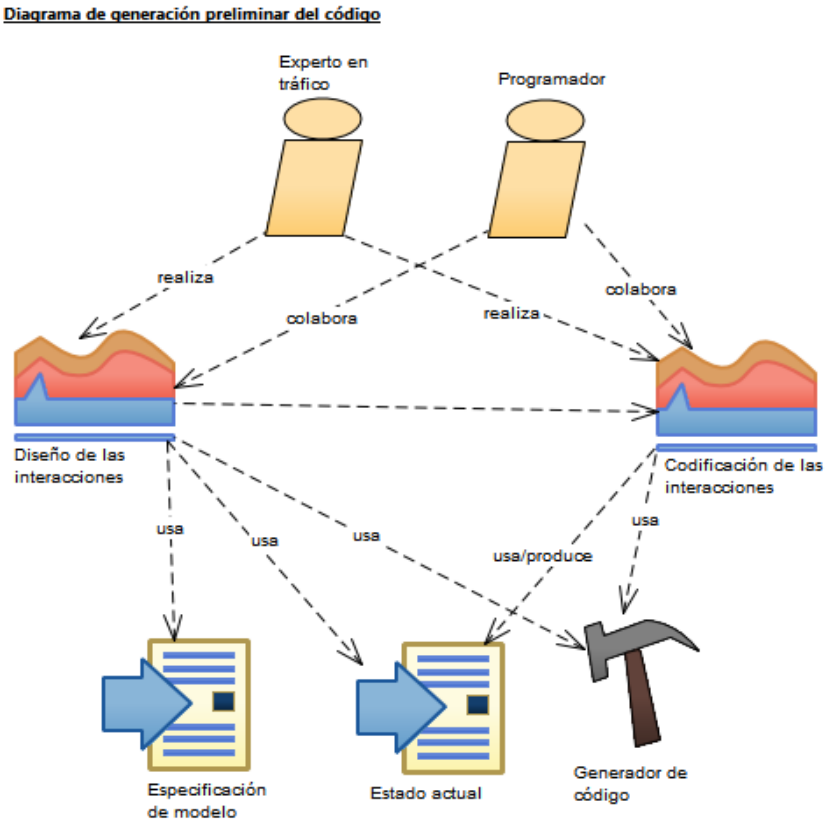

Figura 29: Proceso de generación preliminar de código.

Si la especificación presenta instancias relacionadas con los clústeres *Mental* y *Entorno*, se debe codificar la influencia de cada instancia sobre el resto de instancias de esos clústeres. Dicha codificación permite modificar el comportamiento de los individuos conforme al resultado que se proporcione para cada una de estas instancias relacionadas. Esto es debido a que la toma de decisiones especializada para una plataforma destino se ha de basar en estos resultados. Esta especialización es llevada a cabo en la siguiente fase principal del proceso (ver Sección 5.2.4). Este código se incluye en el cuerpo de los métodos *calculateXValue* de cada clase y el valor obtenido se asigna al atributo *XValues* correspondiente (ver Sección 3.2.1). Por el contrario, si la especificación de modelo dispone de instancias relacionadas con el clúster *Interactivo,* los expertos en tráfico y los programadores colaboran para codificar las instrucciones sobre la toma de decisiones que debe realizarse conforme a las instancias de la especificación. Esto produce un esquema básico en el código sobre cómo las decisiones de los individuos van a ser tomadas (la influencia de los valores de las instancias de los clústeres *Mental* y *Entorno* se introduce en la siguiente fase principal). Así, se deben modificar los métodos *evaluateGoals* presentes en las instancias *Evaluator* creadas y el método *executeChosenTask* contenido en las instancias *Actuator* (ver Sección 3.2.3). El método *calculateSatisfaction* de las instancias *Goal* no es modificado en esta fase principal debido a que la satisfacción de los objetivos de los individuos depende de las instancias presentes en los clústeres *Mental* y *Entorno*.

En la primera fase interna, los expertos en tráfico deben generar el código fuente acorde a la especificación de modelo. Para ello, si en la anterior fase principal se siguió el procedimiento habitual (ver Sección 5.2.2), los expertos deben cargar el archivo XMI con extensión *trafficmodel* en el generador de código. Este archivo contiene el esquema de la especificación de modelo llevada a cabo. Por contra, si se llevó a cabo un procedimiento alternativo (ver Sección 5.2.2), entonces se debe cargar el estado actual producido por el generador de código. En ambos casos, la herramienta genera automáticamente las clases de cada una de las instancias de las que consta la especificación de modelo. Esta generación es transparente al usuario y no necesita soporte por parte de los programadores. Para realizar esta acción el generador de código utiliza las plantillas de código generadas por EMF [85], y el propio archivo XMI o el estado actual del proyecto (contiene su propio archivo XMI). Las plantillas consisten en una clase genérica para cada una de las metaclases presentes en el LMT, considerando los atributos y métodos introducidos en las mismas. Estos métodos presentan cuerpos vacíos o incompletos. A continuación se decide cómo desarrollar las interacciones entre las instancias.

Para las instancias relacionadas con los clústeres *Mental* y *Entorno* se diseña la codificación que se desea aplicar. Esta puede estar basada en algún tipo de teoría o paradigma que haya demostrado su idoneidad en la simulación del comportamiento de los individuos. Una vez desarrollada, los expertos en tráfico y los programadores trabajan conjuntamente para realizar un estudio que evalúe si ésta puede ser codificada de manera factible (por ejemplo, muchas operaciones aritméticas son fácilmente programables). Si el resultado es satisfactorio, se comprueba si las instancias de la especificación de modelo permiten disponer de todos los valores necesarios (por ejemplo, debe poder utilizar objetos de las clases de todas instancias de las que tiene que obtener información) y se hace un análisis de los posibles métodos y atributos

necesarios para llevarla a cabo (por ejemplo, atributos que almacenen valores intermedios o métodos que calculen partes de la fórmula de manera independiente).

En las instancias relacionadas con el clúster *Interactivo* se comprueba cuáles de éstas van a ser utilizadas en cada uno de los métodos a modificar. Este estudio comienza observando cuáles instancias *Goal* están conectadas con cada instancia *Evaluator* en la especificación de modelo. De esta manera se obtiene un esquema sobre qué instancias van a participar en el método *evaluateGoals* de cada instancia *Evaluator*. Esto no es necesario en el caso de que sólo se disponga de una instancia *Evaluator* en la especificación. A continuación, se realiza un análisis similar sobre las instancias *Task* y sus relaciones con las instancias *Actuator* de la especificación. De esta manera se obtiene el esquema de las instancias participantes en el método *executeChosenTask* de cada instancia *Actuator*. De igual forma, si la especificación de modelo sólo presenta una instancia *Actuator* esta parte del estudio puede ser omitida.

Para poder decidir sobre cómo realizar estos diseños, los expertos en tráfico y los programadores pueden comprobar alguna de las implementaciones de especificaciones de modelos que el generador de código presenta como ejemplos o consultar la ayuda del mismo. Una vez completados los análisis oportunos y habiendo decidido la manera de proceder para introducir los distintos cambios en el código, se puede avanzar hacia la siguiente fase interna.

En la segunda fase interna, los programadores introducen el código necesario mediante el editor de texto que incluye el generador de código (ver Sección 4.3). Este código ha de ser introducido en las clases siguiendo la división en clústeres y las relaciones jerárquicas presentes en el LMT (ver Sección 3.2). Así, las primeras clases que se modifican son las que representan instancias del clúster *Entorno*. Una vez completados estos cambios, se modifican las instancias del clúster *Mental*. En ambos casos las clases inicialmente consideradas han de ser las correspondientes a las instancias *XComponent* (ver Sección 3.2) para luego proseguir con las instancias de las metaclases principales. En las primeras, las modificaciones deben comenzar por aquellas que no presenten relaciones de composición y continuar por aquellas que se componen de estas. De esta manera, el programador obtiene una visión global de los cambios que está realizando y se minimiza la posibilidad de errores (por ejemplo, olvidar introducir las modificaciones a una de las clases). Finalmente, si la especificación presenta instancias del clúster *Interactivo* se modifican las clases correspondientes. Estas modificaciones comienzan por las instancias *Actuator* rellenando su método *executeChosenTask* mediante un esquema de código que discrimine entre las instancias *Task* que considera (ver Figura 30). Para realizar esta discriminación hay que preguntar por la instancia *Task* que se está tratando en cada momento. El nombre de esta instancia se indica mediante el atributo *CurrentTask* (ver Sección 3.2.3). A continuación se rellenan los métodos *evaluateGoals* de las clases *Evaluator* de manera similar (ver Figura 31). El nombre de la instancia *Goal* actualmente evaluada se obtiene mediante el atributo *CurrentGoal* (ver Sección 3.2.3).

```
public void executeChosenTask() {

    Boolean found = false;
    Task current = null;
    for (int i=0;i<tExecutes.size() && !found;i++) {
        if (tExecutes.get(i).getTName().equals(currentTask)) {
            found = true;
            current = tExecutes.get(i);
        }
    }
    //do something with current.
}
```

Figura 30: Modificaciones introducidas en las instancias *Actuator*.

```
public void evaluateGoals() {

    Boolean found = false;
    Goal current = null;
    for (int i=0;i<evaluates.size() && !found;i++) {
        if (evaluates.get(i).getGName().equals(currentGoal)) {
            found = true;
            current = evaluates.get(i);
        }
    }
    //do something with current.
}
```

Figura 31: Modificaciones introducidas en las instancias *Evaluator*.

Las clases modificadas son compiladas y almacenadas por la herramienta hasta que se guarde el estado actual del proyecto con los cambios realizados. Esto permite que un estado guardado del proyecto pueda servir como punto de partida para otros. Por ejemplo, permite integrar nuevas interacciones (como nuevas fórmulas de influencia) o completar progresivamente las modificaciones en el código fuente (por ejemplo cuando se considera un conjunto de instancias *Evaluator* y múltiples instancias *Goal* relacionadas). Una vez completada la modificación del código fuente de las clases se puede continuar hacia la siguiente fase principal del proceso de desarrollo. En caso de que apareciesen problemas no detectados con anterioridad durante las modificaciones introducidas en esta fase, se debe volver a la anterior fase interna.

5.2.3.1 Ejemplo de generación preliminar de código

Prosiguiendo con el ejemplo de la sección anterior (ver Sección 5.2.2.1), en esta fase principal se genera el código fuente de la especificación de manera automática. Una vez hecho esto, debido a que la especificación de modelo no presenta instancias relacionadas con el clúster *Interactivo*, se desarrolla y codifica una fórmula para

representar el nivel de influencia de cada instancia de los clústeres *Mental* y *Entorno* sobre el resto de instancias de la especificación de modelo.

En la fase *Diseño de las interacciones* se desarrolla dicha fórmula de influencia y se comprueba su viabilidad para ser codificada. En este caso la fórmula está basada en la lógica borrosa [40], la cual es una extensión de la lógica booleana y ha sido ampliamente utilizada en simulaciones [61]. Así, los atributos *XValues* de las instancias de los clústeres *Mental* y *Entorno* de la especificación de modelo presentarán valores borrosos entre uno y diez según su nivel de influencia sobre el resto. Esta escala de valores permitirá alterar el comportamiento de los individuos de manera simple y controlada.

Para llevar a cabo el desarrollo de la fórmula de influencia se carga inicialmente en el generador de código el archivo XMI que contiene la especificación de modelo diseñada en la anterior fase principal (ver Figura 32). Una vez cargado este archivo, el código fuente de cada una de las clases que representan las instancias de la especificación es generado automáticamente en segundo plano por la herramienta. Navegando a través de las distintas instancias mediante la interfaz gráfica del generador de código, se puede seleccionar cualquiera de ellas para visualizar su código fuente. En este caso, se pueden modificar fácilmente los métodos *calculateXValue* porque su cuerpo aparece originalmente vacío (ver Figura 33), por lo que la aplicación de la fórmula se considera viable. A continuación se especifica la fórmula de influencia. En particular, la fórmula desarrollada proporciona un valor a la instancia actual basado en la suma de los valores del resto de instancias de las que está compuesta y en la división del resultado obtenido por el número de estas. Esto es útil para obtener valores de las instancias principales (por ejemplo *FKnowledge* o *FEnvironment*) relacionados con los valores obtenidos en sus instancias *XComponent*. Una vez realizadas estas estimaciones se puede avanzar hacia la fase interna siguiente.

En la fase interna *Codificación de las interacciones* se procede a implementar mediante código fuente la fórmula desarrollada en la fase interna anterior. Esta codificación comienza modificando las clases de las instancias del clúster *Entorno* para posteriormente continuar con las clases de las instancias del clúster *Mental*. El procedimiento utilizado para llevar a cabo la codificación consiste en modificar inicialmente los métodos *calculateXValue* de las instancias *XComponent* no compuestas hasta llegar a la instancia principal (ver Sección 5.2.3).

En el caso concreto de la instancia *FEnvironment*, el código fuente del cuerpo de su método *calculateEValue* es modificado considerando inicialmente el valor propio de la instancia (almacenado en el atributo *EValues*). Este valor será cargado posteriormente mediante un archivo de configuración en la fase principal *Simulación* del proceso de desarrollo (ver Sección 5.2.5), mientras que dicho archivo será generado en la siguiente fase principal (ver Sección 5.2.4). Una vez obtenido dicho valor, se suman los valores de los atributos *ECValues* del resto de *EComponent* de los que la instancia se compone (por ejemplo, *DayOfWeek* o *TimeOfDay*). A continuación se divide el resultado por la cantidad de instancias *EComponent* (la instancia *FEnvironment* está compuesta de cuatro elementos) más uno, ya que se considera él mismo como elemento a tener en cuenta. Finalmente se aprovecha el encabezado del método *calculateEValue* para retornar el resultado obtenido (ver Figura 34). Este resultado puede ser asignado al atributo *EValues* o utilizado para cualquier otra funcionalidad. Para el resto de instancias el proceso es similar dependiendo del número de instancias *XComponent* de

las que esté compuesta. En el caso concreto en el que una instancia no esté compuesta por otras (por ejemplo, las instancias *Age* o *TripPurpose*), el resultado final será el mismo que el valor inicial de su atributo *XValues* aplicando esta fórmula.

Una vez terminada la codificación de la fórmula en cada una de las instancias y comprobada su correcta compilación, se procede a guardar el estado actual del proceso. Este estado actual almacena las modificaciones realizadas, pudiendo ser reutilizado tanto en esta fase interna como en la anterior gracias a que el archivo XMI que contiene la especificación de modelo también es incluido. Finalmente el proceso de desarrollo avanza hacía la siguiente fase principal.

```
<?xml version="1.0" encoding="UTF-8"?>
<trafficmodel:TrafficModel xmlns:_="" xmlns:trafficmodel="http://trafficmodel/1.0">
  <PDecomposes Drives="#FVehicle" Displays="#FProfile" Possesses="#FKnowledge"
   Id="FPerson"/>
  <EDecomposes _:EName="FEnvironment" EDecomposes="#TimeOfDay
   #DayOfWeek #Weather #RoadCondition"/>
  <KHas KDecomposes="#TripPurpose #LengthOfDrive" KName="FKnowledge"/>
  <PFHas PName="FProfile" PFDecomposes="#IndividualDifferences #Individual"/>
  <VHas Interacts="#FEnvironment" VDecomposes="#Size
   #PerformanceCharateristics" VName="FVehicle"/>
  <VCHas VCName="Size"/>
  <VCHas VCName="PerformanceCharateristics"/>
  <ECHas ECName="TimeOfDay"/>
  <ECHas ECName="DayOfWeek"/>
  <ECHas ECName="Weather"/>
  <ECHas ECName="RoadCondition"/>
  <PFCHas PCName="Age"/>
  <PFCHas PCName="Gender"/>
  <PFCHas PCName="Skills"/>
  <PFCHas PCName="RiskTakingPropersity"/>
  <PFCHas PCName="IndividualDifferences" PFCDecomposes="#Age #Gender
   #Skills #RiskTakingPropersity"/>
  <PFCHas PCName="Individual" PFCDecomposes="#Impairment #Hurry/Distraction"/>
  <PFCHas PCName="Impairment"/>
  <PFCHas PCName="Hurry/Distraction"/>
  <KCHas KCName="TripPurpose"/>
  <KCHas KCName="LengthOfDrive"/>
</trafficmodel:TrafficModel>
```

Figura 32: Contenido del archivo XMI cargado en la herramienta.

```
public int calculateEValue() {
    // TODO: implement this method
    // Ensure that you remove @generated
    // or mark it @generated NOT
    throw new UnsupportedOperationException();
}
```

Figura 33: Método aportado por defecto por la plantilla de EMF.

```
public int calculateEValue() {
    int value = eValues;
    int total = eDecomposes.size() + 1;
    for (int i=0;i<eDecomposes.size();i++)
        value = value + eDecomposes.get(i).getECValues();
    value = value / total;
    return value;
}
```

Figura 34: Método redefinido con la teoría de comportamiento.

5.2.4 Especialización a la plataforma

La cuarta fase principal del proceso de desarrollo introduce las modificaciones necesarias en el código fuente del proyecto (es decir, el estado producido por el generador de código) para conseguir su especialización a la plataforma de simulación destino. Además, en esta fase las especificaciones parciales pueden unirse para formar una especificación de modelo completa (ver Sección 5.2.2) que considere las características de los individuos, su conocimiento actual y la toma de decisiones de los mismos. Para llevar a cabo ambas funcionalidades, esta fase se divide en dos fases internas (ver Figura 35): *Integración de clústeres* e *Implementación de código específico*. Ambas fases están soportadas por el generador de código, permitiendo salvar el estado actual del proceso en cualquier momento. Los roles de usuario implicados en esta fase son el *experto en tráfico* y el *programador*.

La fase interna *Integración de clústeres* sólo es necesaria cuando la especificación de modelo que se está utilizando es parcial. En caso de una especificación completa, esta fase ha de ser omitida, continuando el proceso hacia la siguiente fase interna. En ella los expertos en tráfico utilizan la funcionalidad de integración de clústeres proporcionada por el generador de código (ver Sección 4.3). Esta integración se realiza entre especificaciones parciales que dispongan sólo de instancias relacionadas con los clústeres *Mental* y *Entorno*, y especificaciones que solamente presenten instancias relacionadas con el clúster *Interactivo*. El resultado es una especificación completa (ver Figura 36). El proceso de mezcla es asistido por la herramienta, la cual guía a los usuarios proponiendo las asociaciones apropiadas entre las incluidas en el LMT. Concluido este proceso, se procede a guardar el estado actual y se avanza hacía la siguiente fase interna. En caso de que se produzca algún tipo de error, ambas especificaciones parciales deben de ser revisadas, pudiéndose volver a cualquiera de las fases anteriores del proceso para introducir modificaciones.

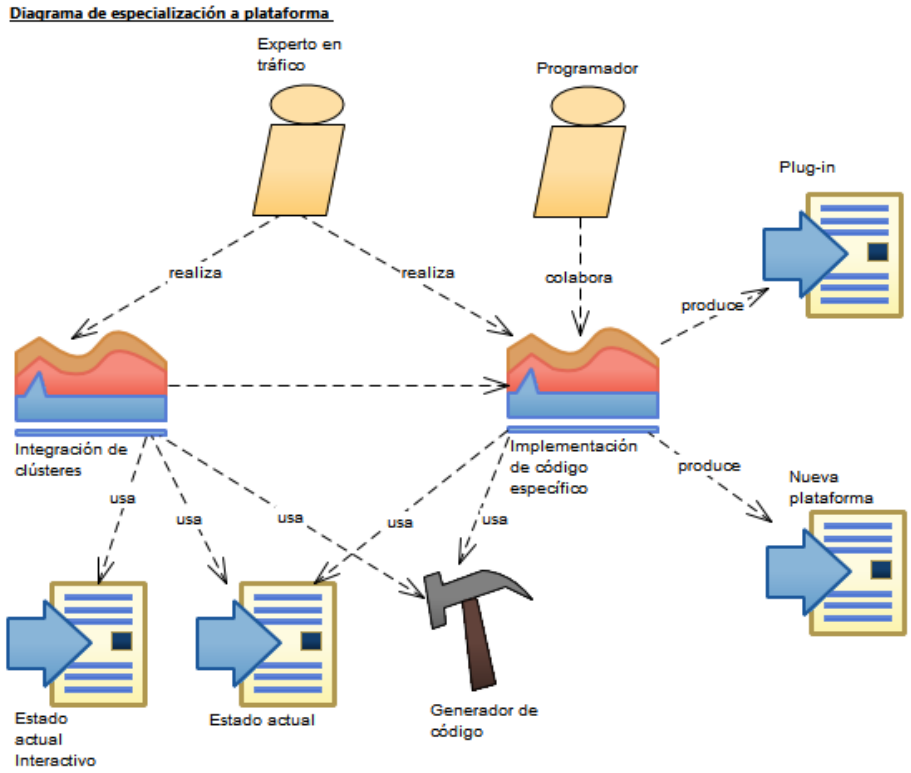

Figura 35: Proceso de creación de una especialización a plataforma.

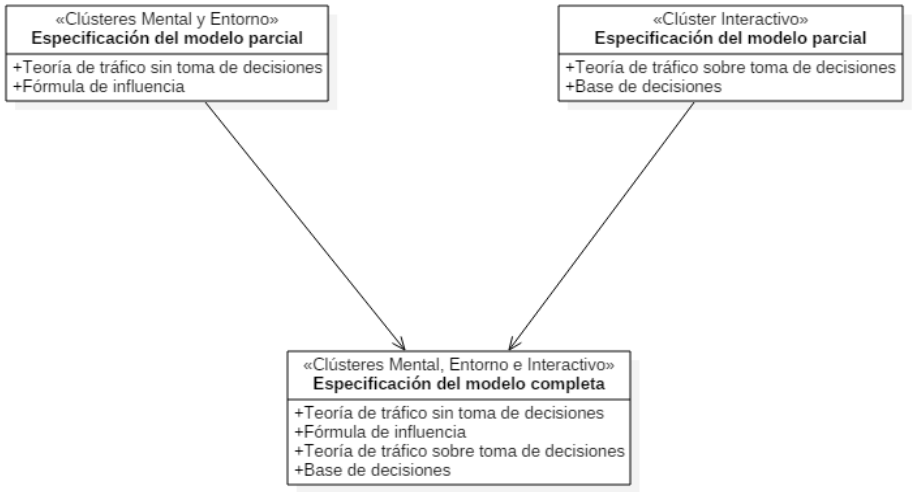

Figura 36: Resultado gráfico de la integración de clústeres.

En la fase interna *Implementación de código específico*, los expertos en tráfico y los programadores colaboran para implementar el código necesario para que una especificación de modelo completa sea compatible con la plataforma destino. Para ello el generador de código proporciona múltiples facilidades. Su editor de texto permite a los usuarios continuar introduciendo modificaciones en el código fuente, mientras que su interfaz gráfica proporciona asistentes para crear clases nuevas pudiendo extenderlas de otras ya existentes en el proyecto.

Los usuarios pueden insertar archivos externos con el propósito de que sean considerados como librerías por el compilador del generador de código. Esto permite que en esta fase puedan ser creados objetos cuyas clases proceden de la plataforma destino, facilitando así la especialización de las clases de los elementos de la especificación de modelo.

Una vez introducidos los cambios necesarios en el proyecto, los usuarios pueden elegir generar dos tipos de archivos comprimidos: un plug-in para la plataforma destino o una nueva plataforma de simulación basada en la plataforma destino original. El primero está enfocado para plataformas en desarrollo, ya que éste puede ser agregado al proyecto permitiendo su utilización como librería. El segundo está pensado para trabajar con plataformas con versiones estables, ya que la nueva plataforma que contiene integra la plataforma original y las teorías de tráfico. Éste puede ser ejecutado directamente para lanzar la plataforma y llevar a cabo simulaciones de tráfico. Ambos archivos contienen la documentación correspondiente asociada al código implementado.

A continuación, el generador de código facilita a los usuarios la creación de un archivo de configuración. Este establece los valores iniciales de las instancias de la especificación y el número de individuos a simular. Para fomentar su legibilidad se codifica en XML. No obstante, este paso puede no ser considerado si la plataforma de destino presenta una interfaz gráfica para configurar sus propios parámetros. Finalmente, si tanto el archivo comprimido resultante como el fichero de configuración han sido generados con éxito por la herramienta, se puede continuar con el proceso de desarrollo avanzando hacia la última fase principal.

5.2.4.1 Ejemplo de especialización a la plataforma

Partiendo del ejemplo considerado en la anterior fase principal (ver Sección 5.2.3.1), en esta fase se toma como referencia la plataforma de test (ver Sección 4.4) y se procede a generar un plug-in directamente insertable en la misma. Además se crea un archivo de configuración para establecer los valores iniciales de *XValue* (ver Sección 3.2.1) de las instancias de la especificación de modelo.

Primero se necesita seleccionar un proyecto desarrollado previamente que contenga una especificación de modelo parcial que considere las interacciones entre los individuos involucrados en el tráfico (es decir, que utilice sólo instancias del clúster *Interactivo*). Esto es debido a que la especificación parcial utilizada en los ejemplos anteriores sólo dispone de instancias de los clústeres *Mental* y *Entorno* para integrar una teoría de tráfico basada en los factores que influyen sobre el comportamiento de los conductores.

El proyecto seleccionado presenta una especificación procedente de una teoría de interacción para los conductores ya probada con anterioridad como parte de este trabajo de tesis [21]. Esta teoría está inspirada en la metodología de agentes INGENIAS (ver

Sección 2.2.4.2), por lo que presenta puntos en común con el clúster *Interactivo* del LMT (ver Figura 37), al tener éste influencias de las metodologías de agentes.

El modelo de comportamiento implícito en la teoría considera que un individuo tiene un objetivo básico con el que se indica su necesidad de llegar lo más rápido posible al destino deseado. Este objetivo está compuesto por otros objetivos mediante descomposiciones en *OR* o en *AND* (ver Figura 38). Estas descomposiciones se interpretan de la siguiente forma: la satisfacción de un objetivo satisface al padre en el primer caso, y sólo la satisfacción de todos los hijos satisface al padre en el segundo. La especificación de modelo utiliza las instancias *Goal* para representar estos objetivos, los cuales tienen al menos una instancia *Task* asociada. Con objeto de distribuir el trabajo de evaluación de las instancias *Goal*, la especificación dispone de una instancia *Evaluator* para cada nivel de descomposición de los objetivos (es decir, cuatro evaluadores). Una instancia *Actuator* se encarga de ejecutar las distintas instancias *Task* (ver Figura 39).

Respecto al código fuente modificado en el cuerpo de los métodos de las instancias del proyecto, éste sólo presenta los cambios propios de la fase *Generación preliminar de código* para una especificación de modelo parcial con instancias del clúster *Interactivo* (ver Sección 5.2.3). Estos cambios afectan a los métodos *evaluateGoals* de las cuatro instancias *Evaluator* y al método *executeChosenTask* de la única instancia *Actuator*. Estas modificaciones permiten considerar las correspondientes instancias *Goal* y *Task* de las que se encarga cada instancia *Evaluator* y *Actuator* respectivamente. Por ello, el proyecto que contiene esta especificación de modelo es suficientemente genérico para ser integrado con otros que contengan especificaciones parciales con instancias solo de los clústeres *Mental* y *Entorno*. En caso de que este proyecto presentase modificaciones adicionales en el código fuente, éstas deberían ser eliminadas previamente a la integración o una vez llevada a cabo ésta.

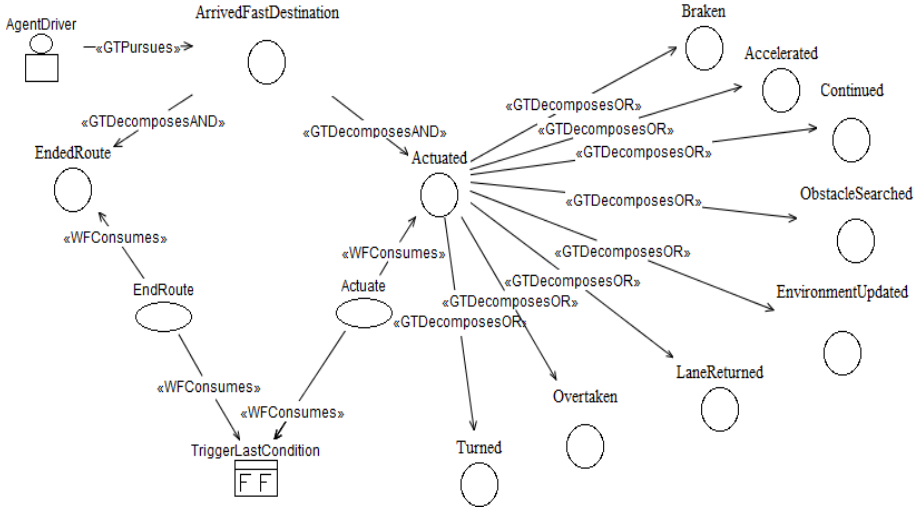

Figura 37: Teoría inicial de interacción de los conductores.

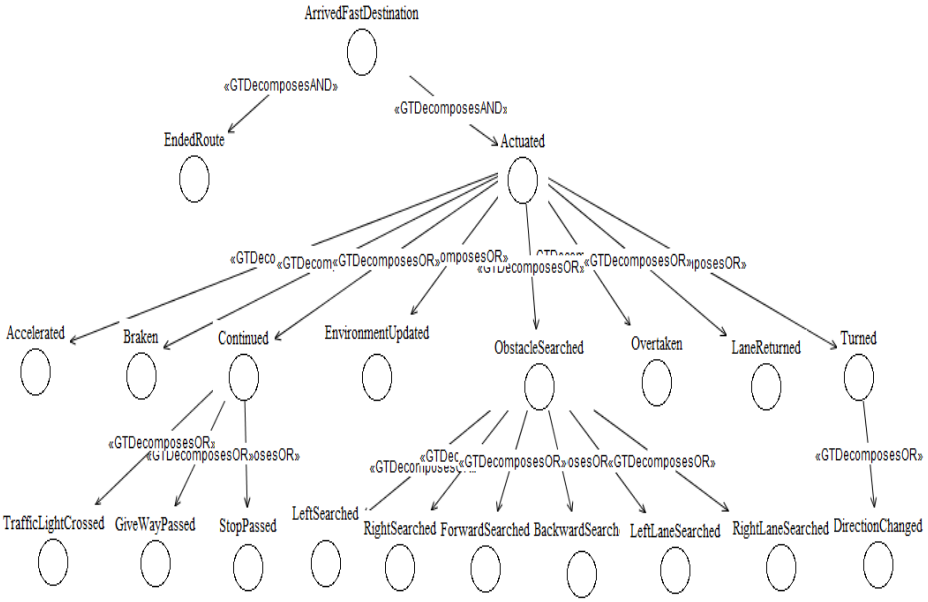

Figura 38: Descomposición de los objetivos del conductor.

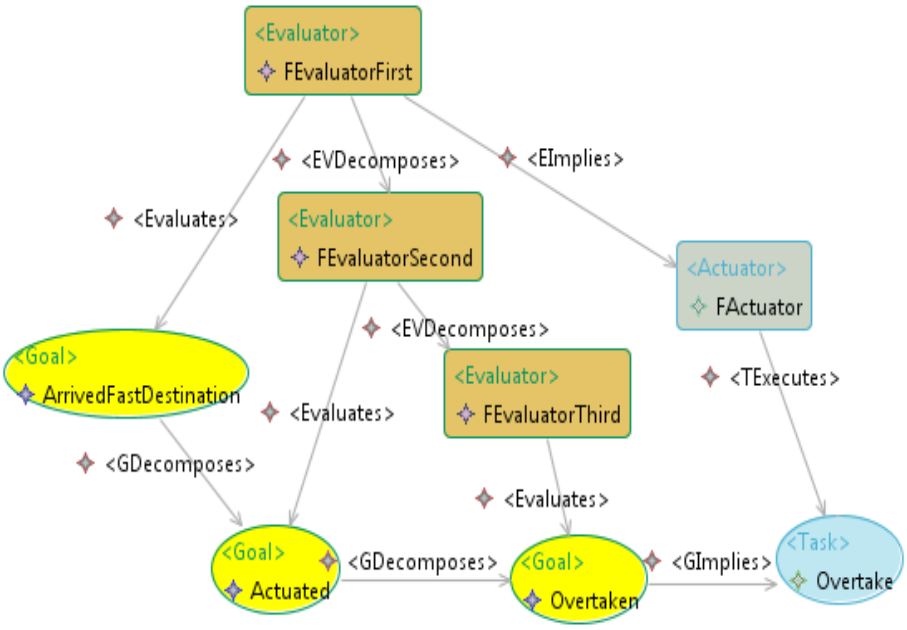

Figura 39: Extracto de la especificación de toma de decisiones.

En la fase interna *Integración de clústeres* se mezclan ambos proyectos utilizando el asistente que proporciona el generador de código para llevar a cabo esta tarea. Este asistente muestra las posibles referencias a utilizar para unir cada instancia de una especificación con la otra. Para ello los expertos deben cargar uno de los dos proyectos en la herramienta y luego mediante el asistente cargar el otro. El orden de carga es indiferente para la correcta ejecución del proceso. A continuación, el proceso comienza uniendo las instancias *Evaluator* con las instancias *Person*. Los usuarios ayudados por el asistente conectan mediante la referencia *IsHarnessed* la instancia *Evaluator FEvaluatorFirst* y la instancia *Person FPerson*, y ambas instancias de manera opuesta a través de la referencia *Harnesses*. Una vez hecho esto la instancia *FPerson* se une a través de la referencia *Pursues* al *Goal ArrivedFastDestination* y mediante la referencia *Utilizes* al *Actuator FActuator*. Estas acciones producen una especificación de modelo completa (ver Figura 40).

En el caso de que ambas especificaciones presenten instancias *Person*, el asistente dispone de un comportamiento por defecto. Este consiste en proponer las instancias *Person* de la especificación de modelo que presente mayor número de ellas. Si ambas especificaciones disponen del mismo número de instancias, entonces el asistente sugiere las instancias de la especificación parcial con clústeres *Mental* y *Entorno*. Estas recomendaciones pueden ser ignoradas por el usuario indicando manualmente las instancias que se desea seleccionar. Una vez concluida la integración entre ambas especificaciones se salva el estado actual de proyecto. Esto permite almacenar dentro del mismo los ficheros fuente y un nuevo archivo XMI con la especificación completa. Finalmente el proceso de desarrollo avanza hacia la nueva fase interna.

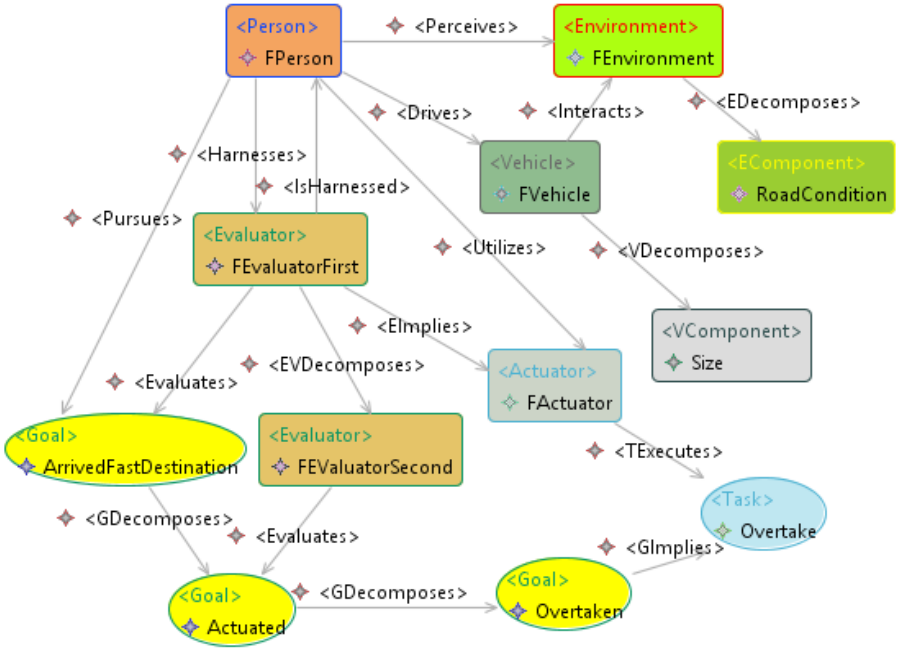

Figura 40: Extracto de la especificación completa.

En la fase *Implementación de código específico* se adaptan los métodos *evaluateGoals* de las cuatro instancias *Evaluator* y el método *executeChosenTask* de la instancia *Actuator* para que consideren los valores producidos por la fórmula de influencia de las instancias de los clústeres *Mental* y *Entorno*. De igual manera se modifican los métodos *calculateSatisfaction* de las instancias *Goal*. Estos métodos han de calcular la satisfacción de cada objetivo diferenciando si presenta una composición en *AND* (ver Figura 41) o en *OR* (ver Figura 42) (por ejemplo, *ArrivedFastDestination* tiene una composición *AND* mientras que *Actuated* se descompone en *OR*). Estos han de actualizar el valor de su correspondiente atributo *Satisfaction* cada vez que sean invocados. Este atributo considera solamente valores verdadero o falso según se haya conseguido o no la satisfacción del objetivo.

No se realizan más cambios en el cuerpo de los métodos debido a que se desea producir un plug-in genérico para ser utilizado como librería en la plataforma de test. En caso de querer llevar a cabo la generación de una nueva plataforma se deberían modificar los métodos *getInstructions* y *setInstructions* de cada instancia *Task* para adaptarlos a la misma, y agregar la plataforma destino original creando y extendiendo clases para abordar la integración de ésta con el proyecto. Una vez completados estos cambios, los expertos en tráfico generan mediante un asistente el plug-in que contiene los ficheros fuente y compilados con los cambios realizados en el código, la documentación del mismo y el fichero XMI con la especificación.

Finalmente, otro asistente guía al usuario en la creación de un archivo de configuración. Este archivo debe contener los valores iniciales de cada una de las instancias relacionadas con los clústeres *Mental* y *Entorno* (ver Figura 43). Si tanto el plug-in producido como el archivo de configuración son ejecutados satisfactoriamente, se procede a continuar hacia la última fase del proceso de desarrollo.

```
public void calculateSatisfaction() {

    Boolean value = true;
    for (int i=0;i<gDecomposes.size() && value;i++)
        value = (Boolean)gDecomposes.get(i).getSatisfaction();
    satisfaction = value;
}
```

Figura 41: Cálculo de satisfacción en composición AND.

```
public void calculateSatisfaction() {

    Boolean value = false;
    for (int i=0;i<gDecomposes.size() && !value;i++)
        value = (Boolean)gDecomposes.get(i).getSatisfaction();
    satisfaction = value;
}
```

Figura 42: Cálculo de satisfacción en composición OR.

```
<?xml version="1.0" encoding="UTF-8"?>
<xml>
    <Drivers>100</Drivers>
    <Pedestrians>0</Pedestrians>
    <Knowledge>
        <name>FKnowledge</name>
        <value>5</value>
    </Knowledge>
    <Environment>
        <name>FEnvironment</name>
        <value>4</value>
    </Environment>
    <Profile>
        <name>FProfile</name>
        <value>2</value>
    </Profile>
    <Vehicle>
        <name>FVehicle</name>
        <value>4</value>
    </Vehicle>
    <KComponent>
        <name>TripPurpose</name>
        <value>7</value>
    </KComponent>
    <KComponent>
        <name>LengthOfDrive</name>
        <value>5</value>
    </KComponent>
     ...
</xml>
```

Figura 43: Fragmento del archivo de configuración de ejemplo.

5.2.5 Simulación

En la quinta fase principal, los expertos en tráfico y los programadores configuran y ejecutan una posible simulación de tráfico en la plataforma destino. Para llevar a cabo dichas tareas esta fase se descompone en dos fases internas (ver Figura 44): *Configuración de simulación* y *Ejecución de simulación*.

En la primera fase interna, la configuración de la simulación puede realizarse de dos maneras distintas. Esto es debido a que en la fase principal anterior se permite producir dos tipos distintos de archivo comprimido (ver Sección 5.2.4). En ambos casos, si se generó un archivo de configuración se pueden introducir cambios sobre los parámetros del mismo para variar las condiciones de la simulación. Una vez realizado esto, si el archivo comprimido contiene una plataforma completa, los expertos pueden continuar el proceso hacia la siguiente fase interna (ya que las modificaciones en los métodos de las clases existentes y la creación de nuevas clases se llevaron a cabo en la anterior fase principal). Por el contrario, si el archivo es un plug-in, los programadores deben

insertarlo en el proyecto de la plataforma destino y añadirlo como librería del mismo. A continuación se deben introducir las modificaciones correspondientes en la plataforma de simulación para que los elementos de la especificación de modelo sean considerados. Estas modificaciones suelen consistir en la generación de nuevas clases o en archivos de configuración o datos.

En la segunda fase interna se ejecuta la simulación propiamente dicha en la plataforma de tráfico destino (plug-in más plataforma o nueva plataforma). Se comprueba que todos los cambios que se han ido introduciendo a lo largo de las fases anteriores funcionen apropiadamente. En caso de que la simulación presente errores o produzca situaciones no esperadas, los usuarios han de volver a cualquiera de las fases previas para llevar a cabo las mejoras oportunas. Si todo funciona apropiadamente, el proceso de desarrollo se ha completado satisfactoriamente.

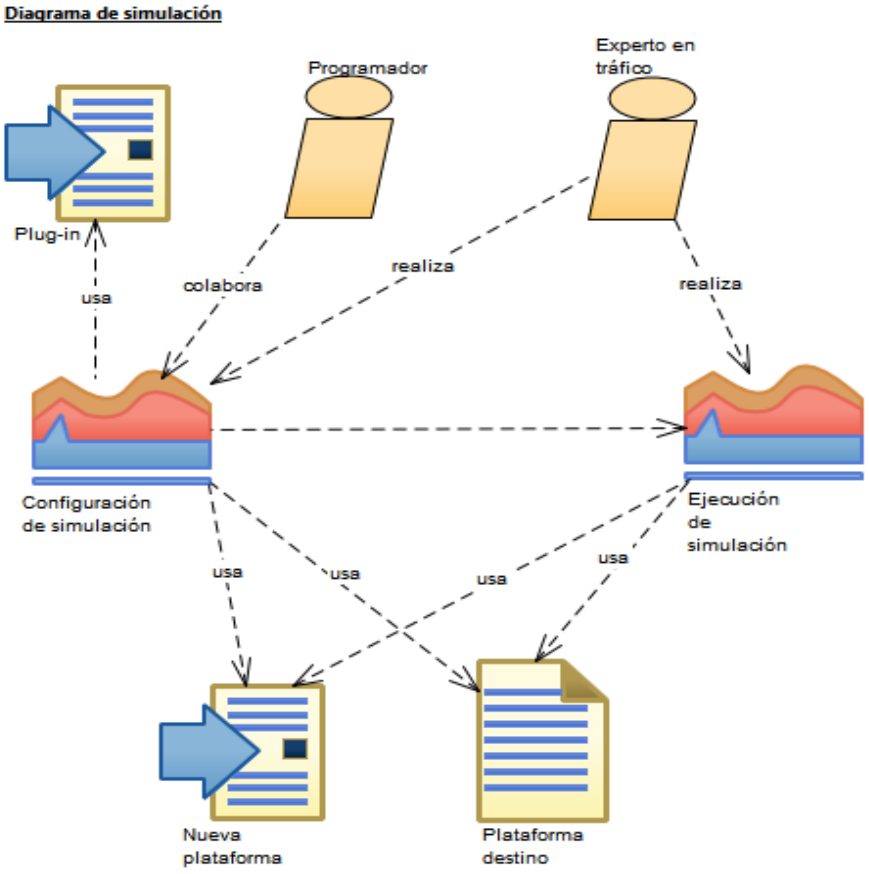

Figura 44: Proceso de simulación con plug-in o nueva plataforma.

5.2.5.1 Ejemplo de simulación

En cuanto al ejemplo, se utiliza el plug-in (ver Sección 5.2.4.1) junto al fichero de configuración generado en la fase principal anterior para ejecutar una simulación de tráfico en la plataforma de test (ver Sección 4.4).

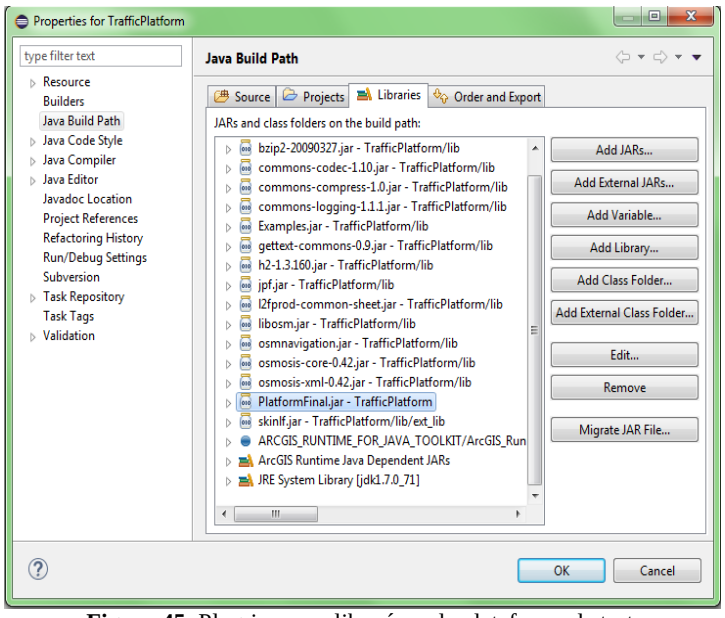

Figura 45: Plug-in como librería en la plataforma de test.

En la fase *Configuración de simulación* se añade este plug-in como librería de la plataforma de test insertándola en la ruta del proyecto (ver Figura 45). A continuación, se crea una nueva clase en el proyecto de la plataforma que se encargue de leer el archivo de configuración y almacenar los parámetros contenidos en él. Después se integra el código proporcionado por el plug-in en las distintas clases de la plataforma. Para ello se procede a generar un nuevo método en la clase que se encarga de ejecutar las simulaciones. Este método debe cargar las instancias de la especificación de modelo y los valores leídos del archivo de configuración. Se deben redefinir los métodos *setInstructions* y *getInstructions* de las instancias *Task* que presenta la especificación de modelo mediante la creación de nuevas clases. Estas clases son extendidas de las clases *Task* proporcionadas por el plug-in. Esto permite realizar las instrucciones correspondientes a cada instancia *Task* en base a los instrumentos proporcionados por la plataforma. También se insertan nuevos atributos de tipo *FPerson* en el constructor de clase de los agentes *Driver* (ver Figura 46). En el cuerpo del método que se encarga de realizar la acción de conducir y continuar con la ruta asignada al individuo, se declaran los cuatro elementos *Evaluator* procedentes de *FPerson* para implementar la toma de decisiones durante la simulación. Esto permite que los individuos no sólo sigan una ruta (comportamiento por defecto), sino que interactúen con el entorno. Esta interacción ha de ser realizada conforme a los factores presentes en la primera teoría de tráfico [68] y a la toma de decisiones introducida en la segunda teoría [21]. Una vez

completados los cambios necesarios en la plataforma de simulación los usuarios pueden avanzar hacia la siguiente fase interna.

Estas modificaciones en la plataforma de test sólo son necesarias la primera vez que se lleva a cabo esta fase interna del proceso. Una vez realizadas, la plataforma necesita modificaciones menores en el código para considerar nuevas instancias procedentes de otros plug-in.

En la fase *Ejecución de simulación* simplemente se ejecuta la plataforma de test y se comprueba que los resultados obtenidos sean los esperados. En caso de producirse errores en la plataforma, los usuarios pueden volver a la anterior fase interna para introducir los cambios en el código de la plataforma que consideren necesarios. Si por el contrario los errores se producen en el código proporcionado por el plug-in, los usuarios deben volver a la fase principal anterior que estimen oportuna y retomar el proceso de desarrollo desde ese punto.

```
1  package agents.individuals;
2
3  import java.util.ArrayList;
9
10 @SuppressWarnings("serial")
11 public class DriverAgentDefault extends DriverAgent {
12
13     private FPerson person;
14
15     public DriverAgentDefault() {
16
17         super();
18         person = new FPerson();
19     }
20
21     public void execDriverAgent(Model model,ArrayList<MapPoint> route) {
22
23         this.model = model;
24         model.getStatistics().setTotalDamount(model.getStatistics().getTotalDamount() + 1);
25         configureDefaultAgent(route);
26     }
27
28     private void configureDefaultAgent(ArrayList<MapPoint> route) {
29
30         vehicle = new Vehicle(Model.Vehicles.CAR.ordinal());
31         resource = new IndividualResource();
32         mentalState.setModel(model);
33         mentalState.setRoute(route);
34         runDefaultAgent();
35     }
```

Figura 46: Creación de objetos del plug-in en la plataforma de test.

5.3 Conclusiones

En este capítulo se ha introducido un proceso de desarrollo de ISDM para simulaciones de tráfico. Se han descrito las fases principales del mismo y las fases internas de cada una de ellas. Estas fases cubren la selección de la base teórica de la simulación, su modelado, la especificación de la propia simulación, y la generación de código fuente, además de la validación de los diferentes artefactos generados.

El proceso de desarrollo se basa en el LMT. Este se centra en el comportamiento de los participantes en el tráfico (adoptando una aproximación de MBA [35]) y sus interacciones (que siguen el modelo CVE [1]).

Dos herramientas apoyan el trabajo con el LMT. Un editor gráfico se utiliza para realizar especificaciones de modelos. El generador de código permite transformaciones de las especificaciones de modelos a código fuente, y la especialización de código para la plataforma de destino. También facilita la reutilización de modelos de la teoría y el código fuente de otros proyectos.

El proceso de desarrollo está concebido como iterativo e incremental, fomentando la reutilización de los artefactos producidos. Además se encarga de refinar los modelos y el código, empezando con aquellos relacionados con los conceptos abstractos (es decir, los del LMT en las especificaciones de modelos y el código original generado por EMF para el LMT) y finalizando con los conceptos del diseño (en el código implementado final).

El proceso tiene dos fases que habitualmente requieren el mayor trabajo en un proyecto. Una es la *Evaluación preliminar de la teoría de tráfico* donde se genera el *planteamiento de modelado*. En ella es donde los expertos de tráfico determinan cómo los elementos de una teoría encajan en el LMT. La segunda es al *Generación preliminar de código*, donde pueden aparecer errores antes de que se logre conseguir la funcionalidad deseada. La fase de *Especialización a la plataforma* suele ser algo menos laboriosa, ya que existe una menor variabilidad entre simulaciones de las plataformas usadas que de las teorías aplicadas. En todos los casos, las herramientas de desarrollo asisten a los usuarios para minimizar estos problemas.

Se ha realizado un ejemplo que produce un plug-in para la plataforma de test (ver Sección 4.4) para mostrar la aplicación de cada una de las fases. Este ejemplo muestra que el proceso de desarrollo realiza una generación de código fuente apropiada para simulaciones en una plataforma de destino dada.

Finalmente, el proceso de desarrollo mantiene cuestiones abiertas, ya que pueden considerarse modificaciones puntuales de la estructura de fases. Además, algunas fases internas podrían ser agrupadas cuando las herramientas de desarrollo simplifiquen el trabajo significativamente. Por ejemplo, la fase interna *Configuración de simulación* podría transformarse para ser integrada en la fase interna *Ejecución de simulación*, simplificando la fase principal del mismo nombre.

Capítulo 6. Experimentación

El marco de desarrollo de ISDM presentado en los capítulos anteriores se ha utilizado para modelar en varios casos de estudio el comportamiento de los individuos involucrados en el tráfico mediante la integración de teorías existentes en el dominio y su implementación en plataformas de simulación. Este capítulo introduce dos de estos casos de estudio que ilustran esta experimentación. Ello, permite evaluar y discutir las ventajas y limitaciones de la aproximación.

6.1 Introducción

Este capítulo recoge dos de los casos de estudio realizados para determinar los requisitos, desarrollar y validar el marco de desarrollo presentado en este trabajo. Se pueden encontrar casos de estudio adicionales en las publicaciones relacionadas con esta tesis (ver Sección 7.3).

El primero de los casos de estudio propone la integración de dos teorías de tráfico y su adaptación a la plataforma de simulación MATSim [90]. Estas teorías son recogidas en dos especificaciones de modelos parciales. La primera utiliza sólo instancias de los clústeres *Mental* y *Entorno* del LMT (ver Secciones 3.2.1 y 3.2.2). Mediante el generador de código (ver Sección 4.3) se crea un nuevo proyecto que genera automáticamente su código fuente asociado. A continuación, se desarrolla y codifica una fórmula de influencia basada en lógica borrosa [61]. La segunda contiene solamente instancias procedentes del clúster *Interactivo* (ver Sección 3.2.3). Esto es debido a que la teoría seleccionada está especializada en la toma de decisiones de los individuos involucrados en el tráfico. De igual manera, se transforma a código fuente mediante el generador de código y se modifica el cuerpo de los métodos de las clases obtenidas. Ambos proyectos se integran utilizando los asistentes de la herramienta, produciendo un nuevo proyecto que engloba la especificación completa. Este proyecto utiliza la plataforma MATSim original como librería. Ello permite especializar el código fuente para generar una nueva versión de la plataforma que considere la gestión de la toma de decisiones en los individuos de sus simulaciones, ya que por defecto MATSim sólo considera la planificación de rutas.

El segundo caso de estudio lleva a cabo la generación de un plug-in específico, el cual es directamente insertable como librería en la plataforma de test (ver Sección 4.4). Para ello, adopta una teoría de tráfico que tiene en cuenta a los peatones y los conductores. Esta teoría produce una especificación de modelo solamente con clústeres *Mental* y *Entorno*. Esto permite reutilizar tanto la fórmula de influencia basada en lógica borrosa como la teoría especializada en la toma de decisiones de los individuos del caso de estudio anterior. Esta última teoría es ampliada para que considere las

decisiones de los peatones junto con las de los conductores. Las especificaciones que modelan ambas teorías son integradas produciendo una nueva especificación del modelo completa. Ésta es utilizada para producir el archivo comprimido que contiene el plug-in. Además, se muestra como la plataforma de test hace uso del plug-in generando objetos presentes en la especificación de modelo.

El resto del capítulo se organiza como sigue. La Sección 6.2 presenta el primer caso de estudio y la 6.3 el segundo. Finalmente, la sección 6.4 presenta las conclusiones obtenidas de los casos de estudio.

6.2 Generación de adaptación completa

Este primer caso de estudio engloba el proceso completo para llevar a cabo simulaciones de tráfico en una plataforma destino partiendo de dos teorías del dominio existentes. Para ello se generan las dos especificaciones de modelos parciales (conformes al LMT) que adaptan las teorías. A continuación, se realiza la transformación a código fuente de ambas, para posteriormente modificar este código de manera independiente de la plataforma de destino. Estas modificaciones persiguen codificar las interacciones básicas de los individuos, ya que el cuerpo de los métodos generados relacionados con este aspecto está inicialmente vacío. Finalmente, se realiza la integración de ambas especificaciones y la especialización del proyecto resultante a la plataforma de simulación de tráfico destino. Ahí se muestra como crear nuevas clases e insertar nuevos fragmentos de código fuente específicos para una plataforma, aquí MATSim [90]. Para ello se siguen las fases del proceso de desarrollo presentado en el capítulo anterior.

El resto de la sección contiene los siguientes apartados. En la Sección 6.2.1 se evalúa la idoneidad de los trabajos seleccionados y se extraen las teorías que presentan. En la Sección 6.2.2 se diseñan las especificaciones correspondientes a dichas teorías. La Sección 6.2.3 describe el desarrollo de las transformaciones a código para dichas especificaciones de modelos. La Sección 6.2.4 aborda la especialización a la plataforma de simulación mediante la creación y extensión de clases. Finalmente, la Sección 6.2.5 introduce el proceso de ejecución de la simulación.

6.2.1 Fase de evaluación preliminar de la teoría

En este caso, el primer trabajo seleccionado describe un modelo llamado *Drivability* [5]. Este modelo describe una serie de aspectos que considera influyentes para la simulación del comportamiento de los conductores.

La *teoría de tráfico* de este trabajo recoge una clasificación de factores organizada en torno a cinco conceptos principales (ver Tabla 3): *Environmental factors*, *Knowledge/Skills*, *Workload*, *Risk awareness* e *Individual resources*. Esta clasificación no presenta elementos relacionados con la toma de decisiones de los conductores. Además, no sigue el enfoque CVE [1] que subyace al LMT, ya que no diferencia los aspectos del entorno y del vehículo. Estos son aunados como elementos del primero. Sin embargo, su adaptación parece factible, debido a que el resto de elementos presentan varios puntos en común con el LMT. Los conceptos principales están compuestos jerárquicamente de manera similar, y aparecen algunos aspectos con el

mismo nombre y semántica. Por ejemplo, *Drivability* introduce características de las personas que pueden ser consideradas por la metaclase *Profile* y sus elementos de composición *PComponent*. También tiene en cuenta factores relacionados con el entorno y con el conocimiento de los individuos, pudiendo ser modelados por las metaclases *Environment* y *Knowlege* junto a sus elementos de composición respectivamente.

Drivability		
Level 1	*Level 2*	*Level 3*
Environmental factors	Vehicle type/status	–
	Traffic hazards	–
	Traffic conditions	–
	Weather conditions	–
	Road conditions	–
	Visibility level	–
Knowledge / Skills	Generic	–
	Self-awareness	–
	Driver training	–
	Driving experience	–
Workload	–	–
Risk awareness	Level of attention	–
	Risk perception	–
	External support	–
Individual resources	Physical condition	Motor Sensoric Organic
	Mental condition	Cognition / Perception Memory Communication Decision
	Socio-Psychological condition	Stress Concentration Reliance Trust Vigilance

Tabla 3: Clasificación de los elementos de *Drivability*.

Una vez estudiada esta teoría de tráfico y su adecuación al LMT se puede confirmar que no hay ningún elemento de la misma que no pueda ser adaptado. Al no presentar aspectos relacionados con la toma de decisiones, se decide realizar una especificación parcial basada en los clústeres *Mental* y *Entorno* (ver Secciones 3.2.1 y 3.2.2 respectivamente) del LMT.

A continuación se comprueba para todas las metaclases principales (por ejemplo, *Knowledge* o *Environment*) de ambos clústeres si éstas disponen de elementos. Así, se observan múltiples aspectos englobados dentro de las metaclases *Profile*, *Knowledge* y

Environment. Por ejemplo, la metaclase *Vehicle* se relaciona con el aspecto *Type/Status* siguiendo el modelo CVE. De esta manera se realiza una clasificación preliminar de los factores para adaptarlos al LMT.

El siguiente paso se centra en la elaboración de un planteamiento de modelado. Este planteamiento se encarga de plasmar la clasificación preliminar realizada con anterioridad. En él se indica cómo los elementos procedentes del modelo de la teoría de tráfico coinciden con las metaclases del metamodelo (ver Tabla 4).

LMT	**Drivability**		
	Level 1	*Level 2*	*Level 3*
Profile	*Workload*	–	–
	Risk awareness	Level of attention	–
		Risk perception	–
		External support	–
	Individual resources	Physical condition	Motor Sensoric Organic
		Mental condition	Cognition / Perception Memory Communication Decision
		Socio-Psychological condition	Stress Concentration Reliance Trust Vigilance
Knowledge	*Knowledge / Skills*	Generic	–
		Self-awareness	–
		Driver training	–
		Driving experience	–
Environment	*Environmental factors*	Traffic hazards	–
		Traffic conditions	–
		Weather conditions	–
		Road conditions	–
		Visibility level	–
Vehicle	*Environmental factors*	Vehicle type/status	–

Tabla 4: Planteamiento de modelado de *Drivability*.

El segundo trabajo presenta una teoría de interacción entre individuos [21] (utilizada previamente en el Capítulo 5 de esta tesis). En el estudio que se lleva a cabo se observa que sólo dispone de elementos relacionados con el clúster *Interactivo*. Se decide realizar una serie de cambios sobre la composición original de los objetivos en ese trabajo con el fin de mostrar la versatilidad de la presente propuesta. La

descomposición original de objetivos ha sido tratada en el ejemplo del capítulo anterior (ver Figura 37).

Los cambios propuestos consisten en suprimir las objetivos *LeftLaneSearched* y *RightLaneSearched* integrándolos en las instancias S*earchedLeft* and *SearchedRight* de manera que se simplifica la composición del objetivo original *SearchedObstacle*, y en añadir una instancia hijo llamada *ContinuedRoute* a la instancia *Continued* para poder continuar avanzado por el entorno manteniendo la velocidad. Esto es debido a que originalmente el objetivo *Continued* sólo era satisfecho cuando se atravesaba un cruce señalizado (por ejemplo, los objetivos *StopPassed* o *TrafficLightCrossed*). En consecuencia cuando el individuo no perseguía los objetivos relacionados con cruces, solamente podía incrementar o disminuir su velocidad mediante los objetivos *Braked* o *Accelerate* para avanzar, sin tener la posibilidad de mantener una velocidad constante.

Actuators	**Evaluators**
DActuator	*EvaluateDestination*
	EvaluateRoute
	EvaluateActions

Tabla 5: Planteamiento de modelado interactivo (parte 1).

Goals			
ArrivedFast Destination	EndedRoute		
	Actuated	Accelerated	
		Braked	
		UpdatedEnvironment	
		ReturnedLane	
		Continued	CrossedStop
			CrossedGiveWay
			CrossedTrafficLight
			ContinuedRoute
		SearchedObstacle	SearchedBackward
			SearchedForward
			SearchedRight
			SearchedLeft
		Overtaken	
		Turned	ChangedDirection

Tabla 6: Planteamiento de modelado interactivo (parte 2).

Tasks			
ArriveFastDestination	EndRoute		
	Actuate	Accelerate	
		Brake	
		UpdateEnvironment	
		ReturnLane	
		Continue	CrossStop
			CrossGiveWay
			CrossTrafficLight
			ContinueRoute
		SearchObstacle	SearchBackward
			SearchForward
			SearchRight
			SearchLeft
		Overtake	
		Turn	ChangeDirection

Tabla 7: Planteamiento de modelado interactivo (parte 3).

A continuación se realiza el planteamiento de modelado. Este planteamiento se basa en los cambios realizados en los objetivos y en la estructura original proporcionada por la teoría. Ésta presenta sólo elementos relacionados con la metaclase *Goal*, por lo que el planteamiento de modelado es completado con los correspondientes elementos *Task* asociados a estos. Además, se añade un elemento *Evaluator* para comprobar que el objetivo primario ha sido satisfecho, otro para revisar el fin de ruta y otro para evaluar los objetivos que llevan a cabo acciones sobre el entorno. Finalmente, se agrega un elemento *Actuator* para ejecutar las instrucciones de los elementos *Task* (ver Tabla 5, Tabla 6 y Tabla 7).

En este caso no se han considerado referencias *GeneralRelationship* entre los elementos. Esto es debido a la necesidad de disponer de una mejor visión del conjunto para estudiar cómo se relacionan los elementos entre ellos. Por tanto su inserción se pospone hasta que ambas especificaciones sean llevadas a cabo.

6.2.2 Fase de diseño de la especificación de modelo

Una vez finalizados los planteamientos de modelado, los elementos que contienen son mapeados a las metaclases seleccionadas correspondientes. Esta acción produce inicialmente un esquema básico como resultado (ver Sección 5.2.2). Éste consiste en una especificación de modelo que sólo contiene las instancias necesarias de las metaclases principales de ambos clústeres (por ejemplo *Person*, *Knowledge* o *Environment*).

En el planteamiento de modelado de *Drivability*, la instancia *DPerson* (una instancia de la metaclase *Person*) es la raíz de la especificación de modelo. Las otras

clases se relacionan con esta clase utilizando las referencias descritas en el metamodelo correspondientes a cada una de ellas. Por ejemplo, la instancia *DEnvironment* es una instancia de la metaclase *Environment* y se relaciona con *DPerson* mediante la referencia *Perceives*, o la clase *DKnowledge* es una instancia de la metaclase *Knowledge* y se relaciona con *DPerson* utilizando la referencia *Possesses*. Este esquema es la base para integrar el resto de la teoría de tráfico con los elementos proporcionados por el metamodelo.

Después de haber creado el esquema, los usuarios proceden a agregar sólo los elementos considerados como representativos de los aspectos descritos en el modelo *Drivability* original. Estos elementos se enlazan con los ya creados siguiendo el planteamiento de modelado.

La instancia *DProfile* actúa como raíz de su propia subestructura arbórea, la cual se descompone en tres *PFComponent* hijos. Estos *PFComponent* representan los elementos de *Drivability Workload*, *IndividualResources* y *RiskAwareness*, ya que todos ellos se han clasificado como características propias de los individuos. *RiskAwareness* presenta tres elementos hijo de forma similar al modelo original: *LevelOfAttention*, *ExternalSupport* y *RiskPerception*. Por su parte, *IndividualResources* es descompuesto en otros tres elementos hijo: *Socio-PsychologicalCondition*, *MentalCondition* y *PhysicalCondition*. El primero a su vez se descompone en: *Stress*, *Reliance*, *Trust*, *Vigilance* y *Concentration*. El segundo lo hace en: *Decision*, *Communication*, *Memory* y *CognitionPerception*. El tercero es descompuesto en: *Motor*, *Sensoric* y *Organic*. Finalmente, *Workload* no presenta elementos hijo. La clase *DEnvironment* es descompuesta en cinco elementos hijo proporcionados por el elemento *Environmental factors* del modelo *Drivability* original: *TrafficHazards*, *WeatherConditions*, *VisibilityLevel*, *TrafficConditions* y *RoadConditions*.

La clase *DVehicle* se compone del elemento *Type/Status* procedente del *Environmental factors* del modelo *Drivability*. Esta diferencia al modelar se debe a que el LMT proporciona una diferenciación entre vehículo y entorno de acuerdo con el enfoque CVE (ver Sección 6.2.1).

La clase *DKnowledge* es conformada por los cuatro elementos hijo originales del modelo *Drivability*: *Generic*, *Driving*E*xperience*, *DriverTraining* y *Self awareness*.

Completado este proceso, la especificación de modelo se ajusta al planteamiento de modelado realizado con anterioridad utilizando solamente las metaclases proporcionadas por los clústeres *Mental* y *Entorno* (ver Figura 47 y Figura 48).

A continuación se estudia la especificación de modelo resultante y se insertan las relaciones de influencia presentes entre los elementos del modelo *Drivability* original. La relación de influencia entre los elementos *Environmental factors* y *Workload* es integrada mediante una instancia de *GeneralRelationship* llamada *EAffectsW* entre las instancias *DEnvironment* y *Workload*. La relación de influencia entre los elementos *Knowledge/Skills* y *Risk awareness* es modelada mediante la relación *KAffectsRA* entre las instancias *DKnowledge* y *RiskAwareness*. Las dos relaciones de influencia que presenta *Individual resources* con *Risk awareness* y *Workload* en el modelo original, son modeladas mediante dos instancias de *GeneralRelationship* llamadas *IRAffectsRA* e *IRAffectsW* respectivamente (ver Figura 49). Por su parte, la relación bidireccional entre los aspectos *Motor* y *Sensoric* es modelada mediante dos relaciones instancias de *GeneralRelationship*. El LMT introduce precisamente esta metaclase para establecer

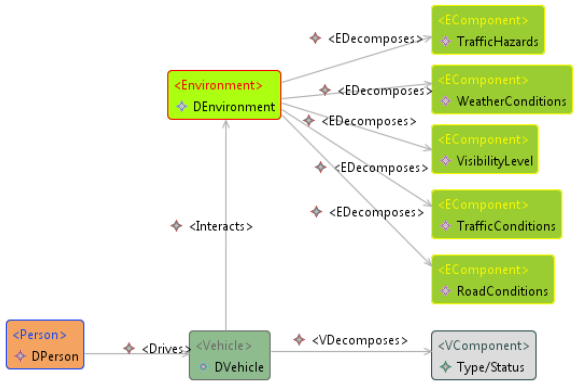

Figura 47: Extracto del clúster *Entorno* para la teoría *Drivability*.

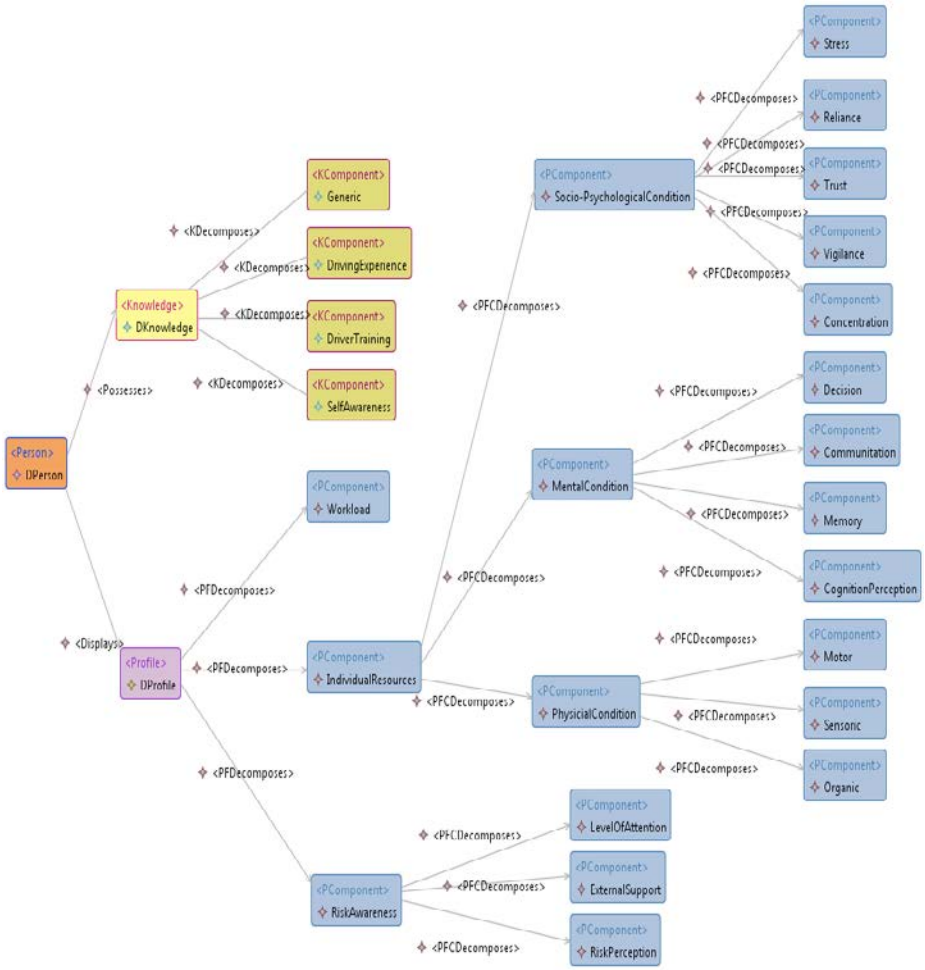

Figura 48: Extracto del clúster *Mental* para la teoría *Drivability*.

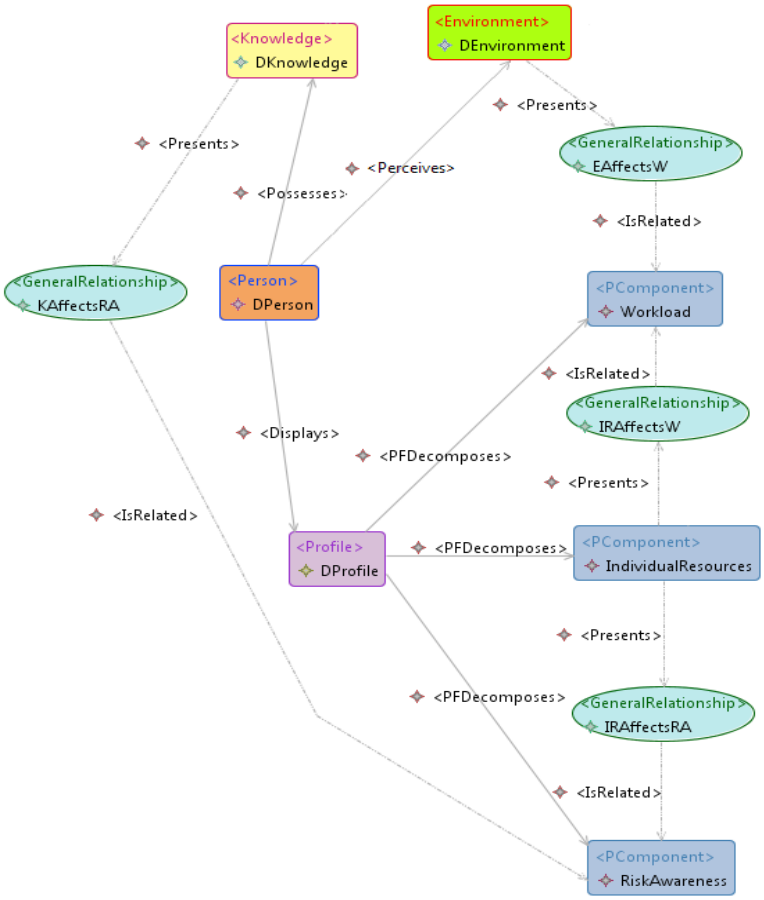

Figura 49: Extracto de la especificación con relaciones.

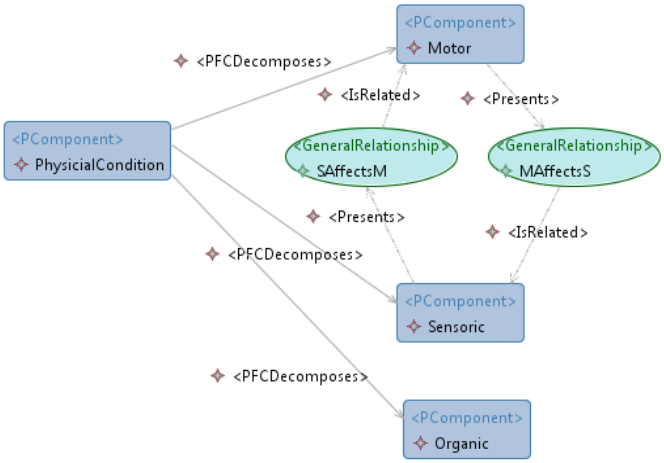

Figura 50: Relación bidireccional entre *Motor* y *Sensoric*.

influencias o colaboraciones entre elementos según las necesidades del modelado. Así, se añade una instancia de *GeneralRelationship* llamada *MAffectsS* para indicar que el elemento *Motor* influye sobre *Sensoric*, y otra instancia llamada *SAffectsM* para señalar que el elemento *Sensoric* influye sobre *Motor* (ver Figura 50).

Una vez que la especificación de modelo basada en los clústeres *Mental* y *Entorno* se ha completado, el siguiente paso es diseñar la especificación de modelo basada en el clúster *Interactivo*. La generación de dicha especificación puede llevarse a cabo con el editor gráfico (ver Sección 4.2) de la manera habitual, o alternativamente con el editor gráfico integrado en el generador de código (ver Sección 4.3). Con el fin de mostrar ambas posibilidades en el diseño de especificaciones se selecciona la segunda opción.

Estudiando el planteamiento de modelado se puede concluir que la teoría original proporciona una estructura de árbol para objetivos y tareas con composiciones *AND* y *OR*. Esta estructura de árbol presenta también las tareas asociadas para el conjunto de los objetivos. Estas tareas se encargan de llevar a cabo las acciones de los individuos siguiendo una serie de instrucciones.

A continuación, se realiza el esquema básico de la especificación utilizando los objetivos clasificados en el planteamiento de modelado. Así, el objetivo raíz llamado *ArrivedFastDestination* representa el objetivo básico de las personas involucradas en el tráfico. Se descompone en dos sub-objetivos que deben cumplirse (es decir, composición *AND*): *Actuated* y *EndedRoute* (ver Figura 51). A su vez, el objetivo *Actuated* se descompone en una serie de objetivos alternativos (es decir, composición *OR*) que representan las diferentes acciones que los conductores pueden llevar a cabo mientras están interactuando en el tráfico (ver Figura 52). Estos objetivos se descomponen en las alternativas para lograrlos (por ejemplo, el objetivo *SearchedObstacle* se descompone en los sub-objetivos *SearchedLeft* o *SearchedForward*). Cuando se cumple alguno de estos objetivos que representan las acciones de los conductores, el objetivo *Actuated* se satisface también. En este momento, el final de la ruta es comprobado (es decir, la satisfacción del objetivo *EndedRoute*). Si el objetivo *EndedRoute* se cumple, entonces el conductor ha logrado su objetivo, ya que se satisface el objetivo raíz *ArrivedFastDestination*; si no, el proceso comienza de nuevo. Esta secuencia de acciones supone que al menos una acción debe llevarse a cabo para satisfacer el objetivo raíz principal.

En esta especificación, las metaclases *Task* y *Goal* usan sus atributos y métodos específicos para gestionar este tipo de composiciones. En las instancias *Goal*, el atributo *GType* indica el tipo de composición (es decir, *AND* u *OR*). Los fragmentos de código son insertados en el cuerpo del método *calculateSatisfaction*. Estos fragmentos de código verifican si las instancias hijas *Goal* son satisfechas. Por su parte, en las instancias *Task*, el atributo *TType* es el encargado de indicar el tipo de composición de la tarea (es decir, *AND* u *OR*), mientras que los fragmentos de código deben ser insertados en el cuerpo del método *setInstructions*. Estos fragmentos de código validan si las instancias hijas *Task* asociadas y las instrucciones atómicas se llevan a cabo con éxito.

Una vez realizado el esquema básico, se continúa el diseño de la especificación de modelo agregando las instancias de las metaclases *Evaluator* y *Actuator*. Estas proporcionan los medios para modelar un ciclo de percepción, razonamiento y actuación [43] similar al que utilizan las personas.

Figura 51: Objetivos raíz con descomposición AND.

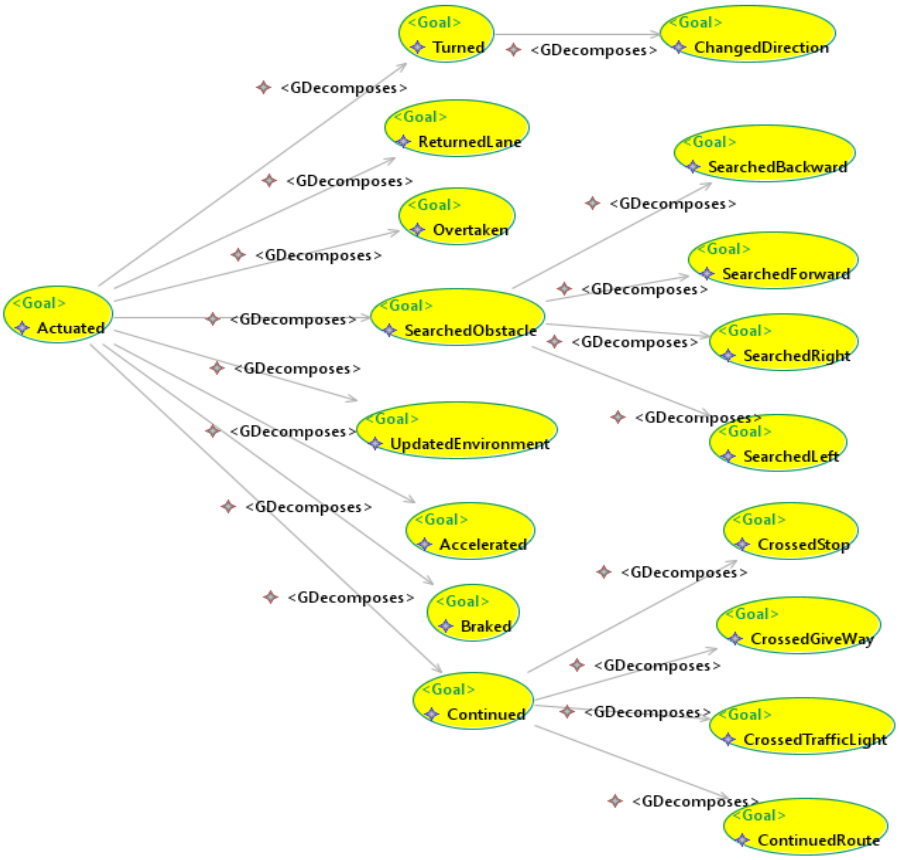

Figura 52: Objetivos de acciones con descomposición OR.

En este caso, los elementos *Evaluator* del planteamiento de modelado tienen una descomposición jerárquica siguiendo el árbol de composición de objetivos. Así, la instancia de *Goal* raíz *ArrivedFastDestination* es considerada sólo por la instancia de *Evaluator* llamada *EvaluateDestination*, el *Goal EndedRoute* es comprobado sólo por la instancia de *Evaluator EvaluateRoute*, y el *Goal Actuated* y el resto de instancias *Goal* relacionadas con las acciones son controladas por la instancia de *Evaluator EvaluateActions*. Este último *Evaluator* se encarga de seleccionar la instancia de *Goal* más adecuada como objetivo del agente. Para ello, cuando la integración de clústeres se lleve a cabo, esta instancia de *Evaluator* tendrá que considerar los parámetros de

entrada proporcionados por los clústeres *Mental* y *Entorno*, o las otras dos instancias de *Evaluator*. También deberá evaluar si se satisface la instancia de *Goal* seleccionada, con el fin de comprobar si sus posibles instancias de *Goal* padres pueden ser satisfechas.

Después de completar la estructura encargada de la evaluación de los objetivos, se añade una instancia de *Actuator* a la especificación de modelo. Este *Actuator* se encarga de todas las instancias de *Task* integradas en la estructura de composición de los objetivos, ejecutando sus instrucciones y sub-tareas cuando la instancia de *Evaluator* apropiada seleccione su objetivo asociado.

Finalmente, no se considera necesario introducir instancias *GeneralRelationship* en la especificación de modelo resultante. Esto es debido a que ninguna instancia del clúster *Interactivo* afecta o influye sobre el resto de manera excepcional.

6.2.3 Fase de generación preliminar de código

Una vez se han generado las dos especificaciones de modelos, se lleva a cabo su transformación a código fuente (ver Sección 5.2.3). Para ello se cargan los ficheros XMI que las contienen en el generador de código. Éste realiza una generación automática basándose en las plantillas de código producidas por EMF y en el LMT.

En el caso de la especificación *Drivability*, tras la carga los expertos navegan por sus elementos para estudiar la fórmula de influencia a aplicar entre sus instancias. Una vez desarrollada la fórmula, se implementa mediante la inserción de los correspondientes fragmentos de código en el cuerpo de los métodos *calculateXValue*. Esto les permite redefinir el procedimiento de cálculo de los atributos *XValues* (ver Sección 3.2). En este caso, los fragmentos de código aplican una fórmula basada en lógica borrosa [40] donde estos atributos presentan valores contenidos entre el 1 y el 10 para determinar su influencia sobre el resto. Dicha fórmula (ver Ecuación 1) ha sido utilizada como ejemplo anteriormente (ver Sección 5.2.3), y aplica el cálculo siguiente: para obtener el valor de un elemento, se suman todos los valores de los elementos hijo junto a su propio valor, dividiendo este resultado por el número de elementos hijo más uno.

$$valor_{actual} = {\sum_{0}^{hijos} valor_{hijo} + valor_{actual}} \Big/ {hijos + 1}$$

Ecuación 1: Fórmula para el cálculo del valor de los elementos.

A continuación se almacena el proyecto resultante utilizando la funcionalidad del generador de código apropiada. Esto permite la reutilización de la fórmula de lógica borrosa, el código fuente y la estructura generada en otros proyectos.

Cuando se carga la especificación basada en el clúster *Interactivo* en el generador de código se produce su código fuente correspondiente. Después, el experto en tráfico y el programador modifican los métodos encargados de llevar a cabo interacciones entre las instancias de manera similar a como se explicó anteriormente (ver Sección 5.2.3). Así, los cuerpos de los métodos *evaluateGoals* de las tres clases *Evaluator* (es decir, *EvaluateDestination*, *EvaluateRoute* y *EvaluateActions*) son modificados de manera genérica permitiéndoles evaluar sólo aquellas instancias *Goal* con las que están relacionadas. De manera análoga, se modifica el método *executeChosenTask* de la clase *Actuator* (es decir, *DActuator*) para que considere todas las instancias *Task* de la

especificación. Finalmente se guarda el proyecto generado por el generador de código salvando el estado actual del mismo.

6.2.4 Fase de especialización a la plataforma

Después de haber almacenado ambos proyectos (el correspondiente a los clústeres *Mental* y *Entorno* por una parte y el del *Interactivo* por otro) mediante archivos comprimidos, se procede a realizar la integración de clústeres. Para esta tarea se usa el asistente del generador de código encargado de esta funcionalidad. Para ello se carga el archivo que contiene la especificación del modelo *Drivability*. A continuación, mediante el asistente de integración se selecciona el proyecto que incluye la especificación con el clúster *Interactivo*. Así, el experto en tráfico vincula la instancia *DPerson* con la instancia de *Goal* raíz de la especificación del modelo (es decir, *ArrivedFastDestination*) a través de la referencia *Pursues*. Entonces, la instancia *DPerson* es conectada a la instancia *Evaluator* raíz (es decir, *EvaluateDestination*) y viceversa, usando las referencias *Harnesses* y *IsHarnessed* respectivamente. Por último, la instancia de *Actuator* se conecta a la instancia *DPerson* por medio de la referencia *Utilizes*. Cuando se completa este proceso y cada referencia queda establecida, ambos grupos se convierten en una única especificación de modelo completa.

El mismo asistente soporta también la definición de las instancias de *GeneralRelationship* entre los elementos de ambas especificaciones de modelos. Estas instancias de *GeneralRelationship* indican la influencia de elementos de origen sobre las instancias *Task* de destino. De esta manera, cuando se ejecuten las instrucciones de la instancia *Task* correspondiente se tendrá en cuenta la influencia del origen.

En particular, la creación de las instancias de *GeneralRelationship* sigue la estructura de los elementos de *Drivability*. Así, los componentes de la instancia *DEnvironment* (es decir, *RoadConditions* o *VisibilityLevel*) afectan directamente a las instancias de *Task Overtake*, *SearchObstacle* o *Brake*, estableciendo múltiples instancias de *GeneralRelationship* entre ellos (ver Figura 53). *RiskAwareness* y sus componentes (es decir, *LevelOfAttention* o *RiskPerception*) afectan a las instancias de *Task Accelerate* o *ReturnLane*. Respecto a la instancia *PhysicalCondition* y sus componentes (es decir, *Sensoric* o *Motor*), estos afectan a las instancias de *Task ObstacleSearch* o *Brake*. Por su parte, el componente *Type/Status* de la instancia *DVehicle* afecta a las instancias de *Task Accelerate, Brake* o *Turn*. En cuanto a los componentes de la instancia *DKnowledge* (es decir, *DrivingExperience* o *DriverTraining*) afectan a las instancias de *Task SearchObstacle*, *Overtake* o *Brake*.

Cuando se completa la integración de estas instancias de *GeneralRelationship*, todos estos conceptos y otros posibles criterios de evaluación deben ser considerados por las instancias *Task*. Para ello se deben insertar fragmentos de código apropiados en el cuerpo del método *setInstructions* de cada una de ellas, ya que este método se encarga de indicar que instrucciones lleva a cabo cada clase *Task*.

A continuación, se usa el generador de código para producir el código fuente específico para la plataforma de destino MATSim. Reflejar el comportamiento del modelo requiere ciertas modificaciones.

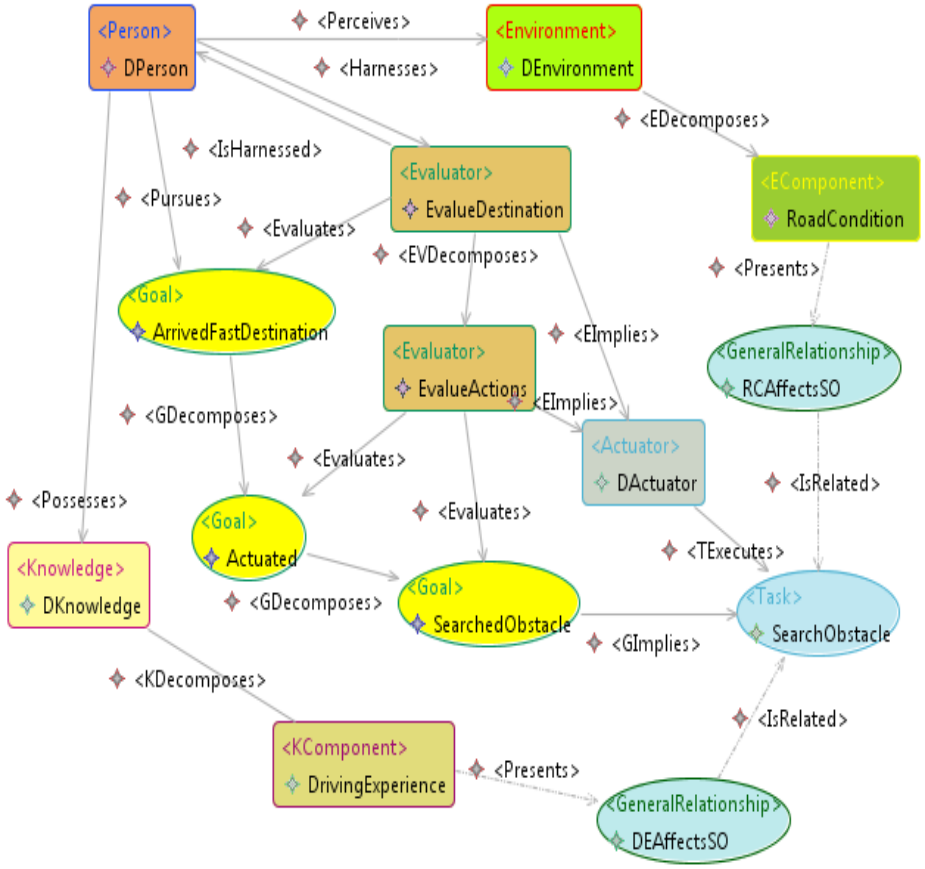

Figura 53: Extracto de la especificación completa con relaciones.

MATSim es una plataforma de agentes. Presenta múltiples funcionalidades y componentes, pero define esencialmente sus individuos (los vehículos) a través de las rutas que han de seguir, soportando ciertas variaciones sobre ellas dependiendo de otros factores. Por tanto, no consideran las interacciones de los individuos [45].

A fin de incorporar las interacciones de los individuos a las simulaciones, se altera el código relacionado. Para ello se carga la plataforma como un archivo comprimido externo junto al resto de sus dependencias (ver Figura 54). Esto permite al generador de código añadirlas cómo librerías dentro del proyecto que contiene la especificación de modelo completa. Así, pueden utilizarse para crear nuevas clases extendiendo las que contienen, o emplear estas últimas para generar objetos en las clases del proyecto.

Antes de modificar las funcionalidades de la plataforma MATSim, se genera un archivo de configuración para proporcionar valores iniciales a las instancias de la especificación *Drivability*. Por ejemplo, aquí se definen los parámetros iniciales de los atributos *XValues*. Estos parámetros se pueden modificar con el fin de obtener diferentes influencias de los elementos. Esta tarea se lleva a cabo mediante un asistente del generador de código.

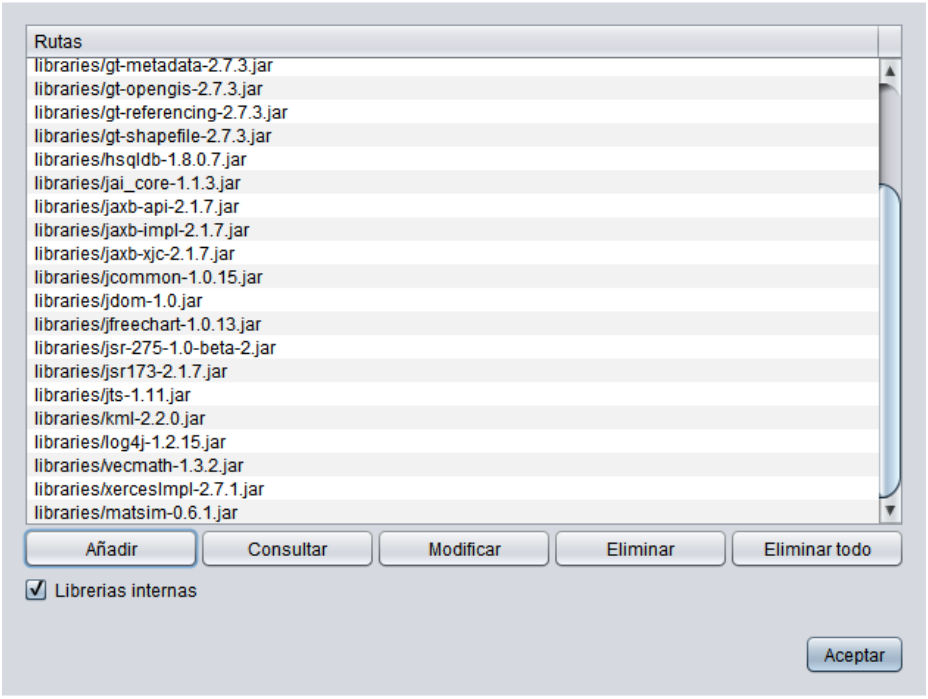

Figura 54: MATSim y sus dependencias cargadas como librerías.

Para conseguir que la plataforma considere la especificación de modelo completa, se requiere el desarrollo y la creación de nuevas clases mediante un asistente de la herramienta. Estas clases son compiladas y almacenadas como parte del proyecto.

En primer lugar, se define una clase para realizar la lectura del archivo de configuración anteriormente producido. Ésta debe cargar los valores de los elementos y alojarlos en la memoria durante la ejecución de las simulaciones.

También se crea una clase que tendrá la responsabilidad de estar a cargo de la conexión entre las clases que corresponden a la especificación de modelo y las de la plataforma. Para ello se utilizan las interfaces externas que ofrecen ambos grupos de clases. De esta forma, las clases que corresponden a la especificación de modelo son encapsuladas en un paquete que puede ser accedido desde el código fuente MATSim.

Por último se desarrolla una nueva clase principal para la plataforma. Ésta debe leer el archivo XMI que contiene la especificación de modelo completa y cargarla en la memoria. También debe utilizar la clase creada anteriormente para leer el fichero de configuración y asignar los valores a las distintas instancias de la especificación. Esto implica que cuando se genere el archivo comprimido resultante, se debe cambiar el manifiesto del mismo para que la plataforma sea lanzada mediante esta clase.

Además de crear las clases anteriores, algunas clases de la plataforma MATSim deben modificarse a fin de considerar la estructura de la especificación de modelo. Las clases originales sólo planean la ruta, y ahora estas clases también deben tener en cuenta, por ejemplo, los adelantamientos o los cambios de carril.

Desde el punto de vista del desarrollo, tanto las nuevas clases como las modificadas se integran en el proyecto de la simulación. De esta forma pueden ser compiladas, almacenadas y utilizadas por el compilador del generador de código.

También el método set*Instructions* de las instancias de *Task* debe volver a ser adaptado y redefinido, ya que encapsula la traducción de las tareas básicas a la plataforma objetivo. En este caso, sus acciones deben adecuarse al modo de funcionamiento e interfaces proporcionadas por MATSim. Estas acciones deben ser asignadas al atributo *Instructions* para que se lleven a cabo cuando la tarea sea ejecutada por la instancia *Actuator*.

Una vez terminada la especialización, se almacena el estado del proyecto. A continuación se realiza la generación del archivo comprimido que contiene la nueva plataforma MATSim modificada. Este fichero es directamente ejecutable.

6.2.5 Fase de simulación

Después de haber realizado las operaciones anteriores se procede a ejecutar la nueva plataforma de simulación MATSim generada. Para ello se necesita el archivo XMI con la especificación de modelo completa y el archivo de configuración creado en la anterior fase. Los valores que contiene este último pueden ser cambiados previamente al lanzamiento de la plataforma para modificar el comportamiento de los individuos que se vayan a simular.

6.3 Generación de plug-in

El segundo caso de estudio describe la utilización de la infraestructura de ISDM con el fin de producir un plug-in directamente insertable como librería en la plataforma de test. Se desarrolla una especificación de modelo que adapta una teoría del dominio basada en el comportamiento de los individuos involucrados en el tráfico (considera peatones y conductores). Además se reutilizan tanto la fórmula de influencia para las instancias de los clústeres *Mental* y *Entorno* como la teoría de toma de decisiones de los conductores introducidas en el caso de estudio anterior (ver Sección 6.2). Ésta última es ampliada introduciendo otra estructura equivalente para los peatones.

La descripción de este caso de estudio se organiza en las siguientes secciones. En la Sección 6.3.1 se estudian y modifican las aproximaciones del dominio seleccionadas y se comprueban sus teorías. La Sección 6.3.2 aborda el diseño de las especificaciones de modelos basándose en estas teorías. En la Sección 6.3.3 se realizan las transformaciones a código. Luego, la Sección 6.3.4 cubre la integración de clústeres y la especialización a la plataforma de test. Finalmente, en la Sección 6.3.5 se modifica la plataforma para poder ejecutar simulaciones que consideren las teorías estudiadas.

6.3.1 Fase de evaluación preliminar de la teoría

El trabajo de tráfico seleccionado [78] presenta una teoría que describe una serie de factores de riesgo que influyen en la siniestralidad entre peatones jóvenes. Estos permiten clasificar ciertas características de los mismos (ver Tabla 8), de los conductores con los que interactúan (ver Tabla 9) y del entorno (ver Tabla 10) de

manera jerárquica. Esta clasificación jerárquica permite realizar una taxonomía donde ciertos factores dependen o se ven influidos por otros.

Factors related to the child			
Stage of development	*Anatomic development*	*Social environment*	
		Effect of others	**Family factors**
Attention span	Stature	Crossing guards	Poverty
Walking speed	Reflective clothes	Accompanying adult	Crowding
Midblock dart-outs	Sudden appearance	Peers	Care of local authorities

Tabla 8: Simplificación de los factores de los peatones (parte 1).

Factors related to the driver
Children in the road
Use of alcohol
Driver fatigue

Tabla 9: Simplificación de los factores de los peatones (parte 2).

Factors related to the environment			
Type of road	*Layout of road*	*Traffic signals and islands*	*Traffic conditions*
Driveway			Traffic density and vehicle speed
Parking lot			Neighborhood characteristics
Freeway			Roadside parking
Arterial road			Vehicle design
Collector road			Inclement Weather
Local street			Darkness

Tabla 10: Simplificación de los factores de los peatones (parte 3).

La teoría de tráfico no considera el enfoque CVE integrado en el LMT, por lo que hay que determinar la forma en que se pueden insertar los factores en la especificación de modelo como componentes de las metaclases principales. De nuevo también, en esta teoría no aparecen elementos relacionados con la toma de decisiones de los individuos, sino solamente factores de riesgo que pueden ser descritos por los clústeres *Mental* y *Entorno* del LMT (ver Secciones 3.2.1 y 3.2.2) sin hacer uso del clúster *Interactivo* (ver Sección 3.2.3). Por ello, también es necesario seleccionar una teoría de tráfico enfocada en la toma de decisiones para poder generar una especificación de modelo completa.

El primer paso para modelar la teoría basada en los factores de riesgo de los peatones jóvenes consiste en la elaboración de un planteamiento de modelado. Este planteamiento se lleva a cabo con el objetivo de buscar equivalencias entre los diferentes factores considerados en la teoría y los conceptos proporcionados por el LMT. Debido a que los nombres de los factores utilizados son largos en el trabajo original, aquí se han simplificado para facilitar su visibilidad en los diagramas.

Se desarrollan cuatro grupos principales de conceptos: *Pedestrian Profile* (*PProfile*), *Environment*, *Vehicle* y *Driver Profile* (*DProfile*). El primer grupo engloba todos los elementos relacionados con las características de los peatones jóvenes (por ejemplo, *Famility factors* o *Anatomic development*) (ver Tabla 11). El segundo grupo abarca los factores presentes en el entorno (por ejemplo, *Layout of road* o *Type of road*) (ver Tabla 12). Estos factores son comunes tanto a los conductores como a los peatones. El tercer grupo describe los factores propios de los vehículos (por ejemplo, *Design* o *Speed*) (ver Tabla 13). Como el LMT se basa en el enfoque CVE, estos factores no se consideran vinculados al entorno, sino parte de una instancia *Vehicle*. Finalmente, el cuarto grupo considera las características de los conductores (por ejemplo, *Driver fatigue* o *Use of alcohol*) (ver Tabla 14). Se puede observar que tanto en el caso de los peatones como de los conductores, no es necesaria la utilización de los conceptos relacionados con *Knowledge*. Los factores de riesgo se aplican a características de las personas, del vehículo o del entorno, y no se consideran elementos basados en el conocimiento de los individuos.

En cuanto a la teoría de interacción de los peatones, se reutiliza la teoría de toma de decisiones para los conductores desarrollada anteriormente [21] (ver Sección 6.2), aunque adaptándola a las necesidades específicas de la plataforma de test. Esta teoría se extiende para que soporte la toma de decisiones de los peatones mediante las consideraciones indicadas en [25]. Éstas introducen las diferentes acciones que llevan a cabo los peatones cuando interactúan con el resto de elementos del tráfico. Así se considera que ambos tipos de individuos (conductores y peatones) tienen muchos objetivos comunes (por ejemplo, ambos buscan obstáculos, aceleran o pueden cambiar de sentido). Por lo tanto, la teoría revisada para ambos tipos de personas se basa también en una estructura arbórea de instancias *Goal* con descomposiciones *AND* y *OR* con cuatro niveles de profundidad. No obstante, los objetivos se vinculan a tareas diferentes y condiciones diferentes (por ejemplo, un peatón no adelanta a otro de la misma forma que dos conductores). Por este motivo se duplica la estructura de objetivos y para evitar solapamientos en los nombres de las nuevas instancias de *Goal*, estos se preceden de una *P* que indica que corresponden a *Pedestrians*.

Pedestrian Profile		
Family factors	*State of development*	*Anatomic development*
Poverty	Attention span	Stature
Crowding	Walking Speed	Reflective clothes
Care of Local Authorities	Midblock dart-outs	Sudden appearance

Tabla 11: Planteamiento del *Profile* de los peatones.

Environment				
Adverse traffic conditions	*Traffic signals and islands*	*Layout of road*	*Type of road*	*Social Environment*
Darkness				**Effects of others**
Inclement weather				Crossing guards
Roadside parking				Accompanying adult
Traffic density				Peers
Neighborhood characteristics				

Tabla 12: Planteamiento de los factores *Environment*.

Vehicle
Design
Speed

Tabla 13: Planteamiento del *Vehicle* de los conductores.

Driver Profile
Driver fatigue
Use of alcohol
Children in the road

Tabla 14: Planteamiento del *Profile* de los conductores.

Actuators	**Evaluators**
PActuator	*EvaluatorFirst*
	EvaluatorSecond
	EvaluatorThird
	EvaluatorFourth

Tabla 15: Planteamiento de la toma de decisiones de los peatones (parte 1).

Goals			
PArrived Fast Destination	PEndedRoute		
	PActuated	PAccelerated	
		PSlowed	
		PUpdatedEnvironment	
		PMoved	PPassedCrossing
			PPassedWay
			PContinuedPath
		PSearchedObstacle	PSearchedBackward
			PSearchedForward
			PSearchedRight
			PSearchedLeft
		PDodged	PDodgedRight
			PDodgedLeft
		PTurned	PChangedDirection

Tabla 16: Planteamiento de la toma de decisiones de los peatones (parte 2).

Tasks			
PArriveFast Destination	PEndRoute		
	PActuate	PAccelerate	
		PSlow	
		PUpdateEnvironment	
		PMove	PPassCrossing
			PPassWay
			PContinuePath
		PSearchObstacle	PSearchBackward
			PSearchForward
			PSearchRight
			PSearchLeft
		PDodge	PDodgeRight
			PDodgeLeft
		PTurn	PChangeDirection

Tabla 17: Planteamiento de la toma de decisiones de los peatones (parte 3).

En el planteamiento de modelado se identifican estos elementos *Goal* (ver Tabla 16) y los elementos *Task* asociados a estos (ver Tabla 17). Se añade un elemento *Actuator* para la ejecución de estos últimos. Finalmente, se decide introducir cuatro elementos *Evaluator* (ver Tabla 15) para la evaluación de los elementos *Goal* (uno por cada nivel de composición que presenta el árbol de objetivos).

Al igual que en anterior caso de estudio (ver Sección 6.2.1), aquí no se han considerado referencias *GeneralRelationship* entre los elementos en esta fase. Esto es debido a la gran cantidad de elementos que presentan las especificaciones de modelos que están siendo tratadas, y en consecuencia a la necesidad de disponer de una mejor visión global de las instancias y relaciones. Esta visión global se obtiene en fases posteriores, al diseñar las correspondientes especificaciones de modelos mediante el editor gráfico. Entonces se pueden analizar de una forma global las relaciones existentes y la posible necesidad de introducir nuevas instancias de *GeneralRelationship*.

6.3.2 Fase de diseño de la especificación de modelo

Una vez desarrollados los planteamientos de modelado de ambas teorías, los elementos principales de cada grupo son mapeados a instancias de las metaclases proporcionadas por el LMT. De esta manera se realizan las especificaciones de modelos.

En el caso de la teoría de factores de riesgo de los peatones, se utilizan solamente las metaclases correspondientes a los clústeres *Mental* y *Entorno* (ver Secciones 3.2.1 y 3.2.2 respectivamente). Creando instancias de las mismas se obtiene un esquema básico. Así, se crean dos instancias *Person*, *Pedestrian* y *Driver*, para representar los distintos tipos de conductores y peatones respectivamente. Estas instancias son la base de esta especificación de modelo y el resto de clases dependerán de ellas. Cada una de

estas instancias tiene su propia instancia *Profile* relacionada mediante la referencia *Displays*: *DProfile* para la instancia *Driver* y *PProfile* para la instancia *Pedestrian*. Sin embargo, ambas comparten una única instancia *Environment* llamada *GEnvironment*. La instancia *Pedestrian* se relaciona directamente con *GEnvironment* mediante la referencia *Perceives*, mientras que la instancia *Driver* se conecta con *GEnvironment* a través de una instancia *Vehicle* llamada *DVehicle* que utiliza una referencia *Interacts* (ver Figura 55).

Con los elementos anteriores se completa el esquema básico. A continuación se procede a insertar el resto de elementos según las clasificaciones realizadas en el planteamiento de modelado.

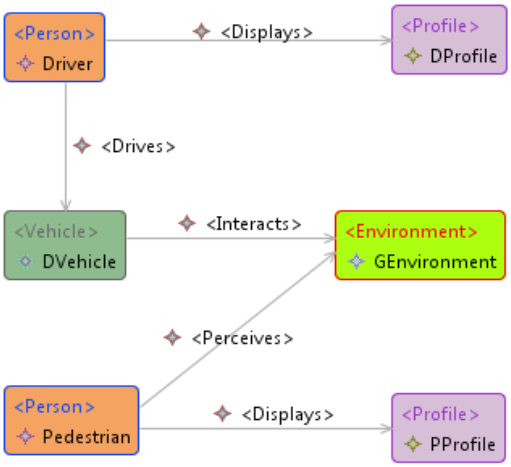

Figura 55: Esquema inicial con los elementos principales.

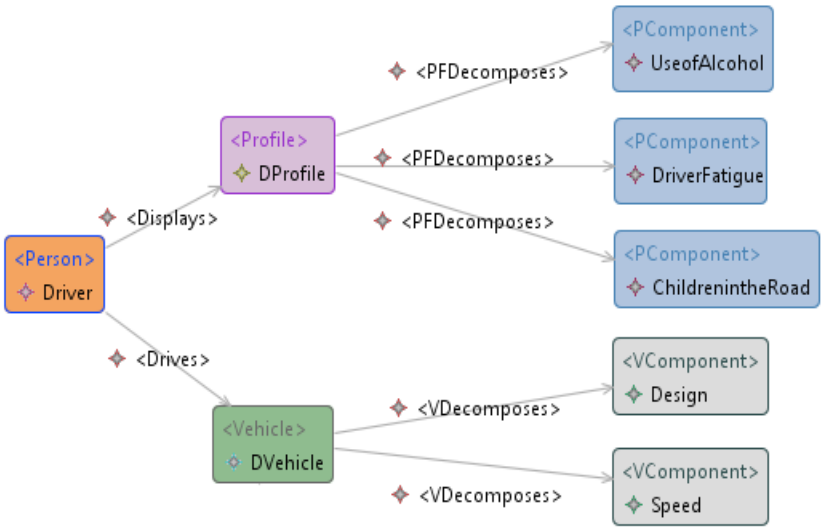

Figura 56: Extracto de la especificación de la instancia *Driver*.

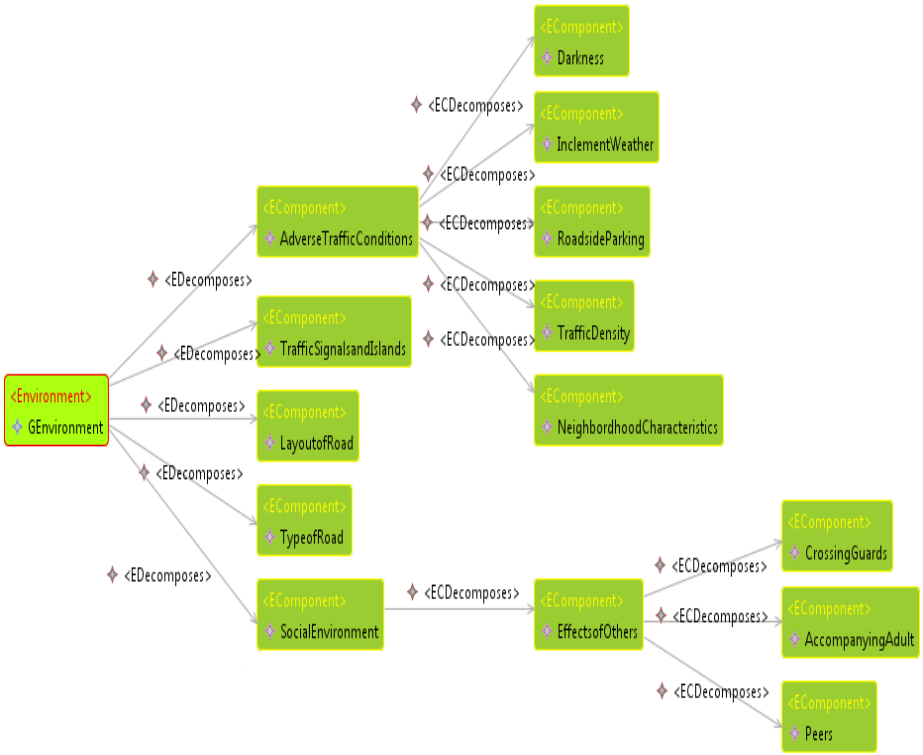

Figura 57: Extracto de la especificación de *GEnvironment*.

De esta manera, la instancia de *Driver* tiene dos instancias relacionadas *DProfile* y *DVehicle*. La primera presenta tres instancias de *PFComponent* como hijos llamadas *UseofAlcohol*, *DriverFatigue* y *ChildrenintheRoad*. De acuerdo con la teoría original, estas cubren las características de los conductores que ponen en riesgo a los peatones. La segunda se compone de dos instancias de *VComponent* hijos llamadas *Design* y *Speed*. Este último es extraído de la división en dos factores nuevos del atributo de la teoría original llamado *Traffic density and speed*. Esto es necesario porque el LMT utiliza el enfoque CVE. Así, el atributo de densidad del tráfico es considerado como parte del entorno de manera general (ver Figura 56), mientras que la velocidad lo es en los vehículos.

La instancia *GEnvironment* es la raíz del conjunto de elementos que establecen los factores ambientales que influyen tanto en los conductores como en los peatones. Se descompone en cinco instancias de *EComponent* obtenidas del planteamiento de modelado: *AdverseTrafficConditions*, *TrafficSignalsandIslands*, *LayoutofRoad*, *TypeofRoad* y *SocialEnvironment* (ver Figura 57). Esta última instancia, aunque originalmente forma parte de los factores relacionados con los peatones jóvenes, ha sido reclasificada como componente del entorno en el planteamiento de modelado. Originalmente se descompone en los factores *Effects of others* y *Family factors*. Sin embargo, en el planteamiento de modelado la instancia *SocialEnvironment* sólo se descompone en la instancia *EffectofOthers*. La instancia *FamilyFactors* permanece como característica propia de los peatones, formando parte de la instancia *PProfile*.

Esta instancia *EffectofOthers* está compuesta por tres instancias de *EComponent*: *CrossingGuards*, *AccompanyingAdult* y *Peers*. En cuanto a la instancia *AdverseTrafficConditions*, se descompone en otras cinco instancias: *Darkness*, *InclementWeather*, *RoadsideParking*, *TrafficDensity* y *NeightbordhoodCharacteristics*. Indicar también que la instancia *TypeofRoad* ya considera por sí misma los elementos específicos presentes en la teoría de tráfico (es decir, *Driveway* o *Local street*), por lo que no ha sido descompuesta en nuevas instancias de *EComponent*.

Finalmente, la instancia *PProfile* está encargada de modelar las características propias de los peatones jóvenes. Se descompone en tres instancias siguiendo la clasificación original de la teoría: *AnatomicDevelopment*, *StateofDevelopment* y la instancia anteriormente introducida *FamilyFactors*. La primera está compuesta a su vez por tres instancias hijo: *Stature*, *ReflectiveClothes* y *SuddenApearance*. La segunda se compone también de tres instancias *PComponent* llamadas *AttentionSpan*, *WalkingSpeed* y *MidblockDart-outs*. La tercera instancia está compuesta por otras tres instancias, llmadas *Poverty*, *Crowding* y *CareofLocalAuthorities* (ver Figura 58).

Tras completar la especificación de modelo con los conceptos de los clústeres *Mental* y *Entorno*, se procede a considerar las posibles relaciones de influencia entre estos elementos mediante instancias de la metaclase *GeneralRelationship*. Como se indicó anteriormente (ver Sección 6.3.1) estas relaciones de influencia pueden diseñarse en el planteamiento de modelado, pero se ha preferido esperar a completar la inserción de todos los elementos de la especificación de modelo.

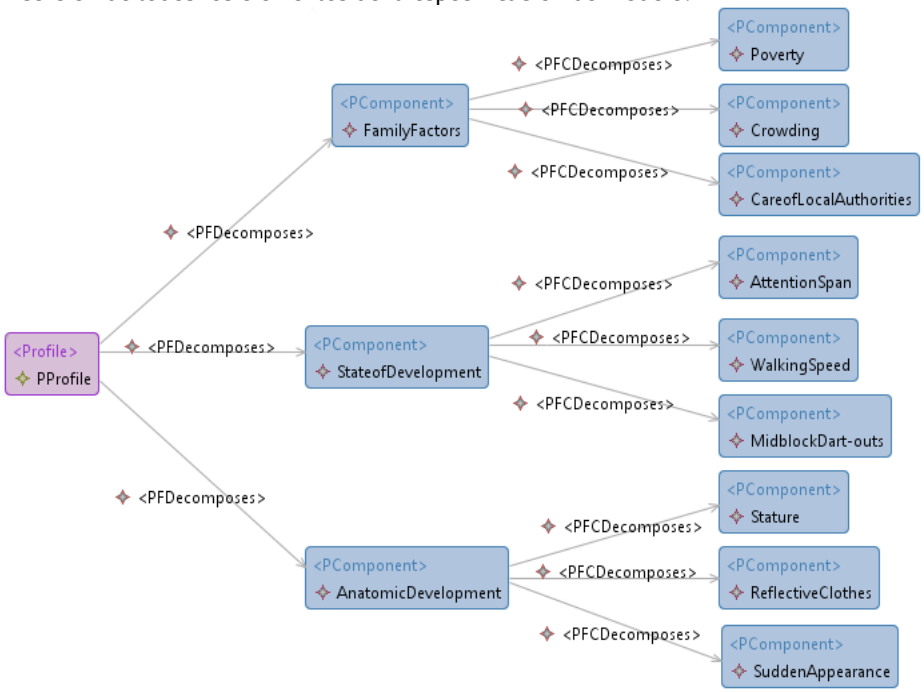

Figura 58: Extracto de la especificación de la instancia *PProfile*.

En este caso, se insertan dos instancias de *GeneralRelationship* con sentidos opuestos entre los elementos *SocialEnvironment* y *FamilyFactors*, llamadas *SEAffectsFA* y *FAAffectsSE* (ver Figura 59). Esto es debido a que la teoría considera a ambos como factores del entorno, mientras que en la especificación de modelo el segundo ha sido reubicado dentro de las características individuales de las personas. Además, se introduce otra *GeneralRelationship* entre las instancias *Speed* y *TrafficDensity* para indicar la influencia de la segunda sobre la primera. Como se indicó anteriormente, en la teoría original estos dos factores estaban considerados por un único factor (ver Figura 60).

Una vez concluida la especificación de modelo que utiliza los clústeres *Mental* y *Entorno*, se procede a diseñar la especificación de modelo que utiliza solamente los conceptos del clúster *Interactivo* y las *GeneralRelationship*. De la misma manera que en el caso anterior, y para mostrar la capacidad de reutilización, este procedimiento se lleva a cabo mediante el editor gráfico integrado en el generador de código. Esto permite continuar con la teoría de interacción de los conductores ya modelada e insertar los nuevos elementos relacionados con los peatones.

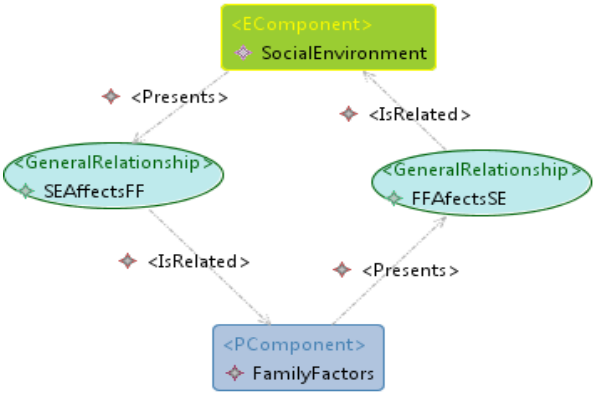

Figura 59: Relaciones entre *FamilyFactors* y *SocialEnvironment*.

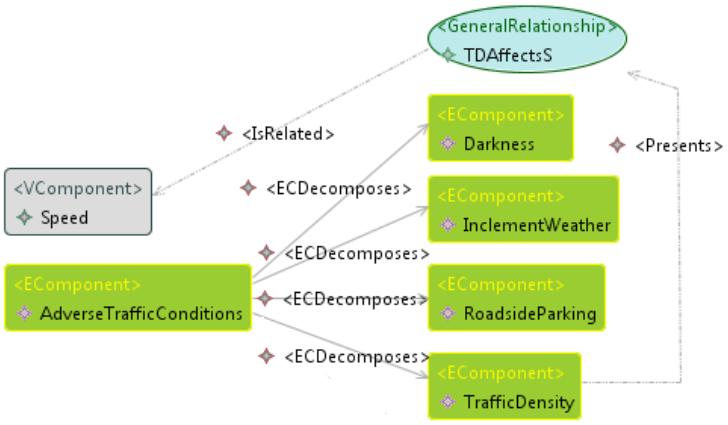

Figura 60: Relación entre *TrafficDensity* y *Speed*.

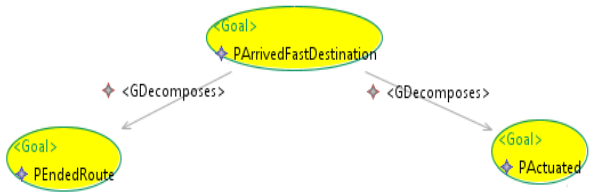

Figura 61: Objetivos raíz de los peatones con composición AND.

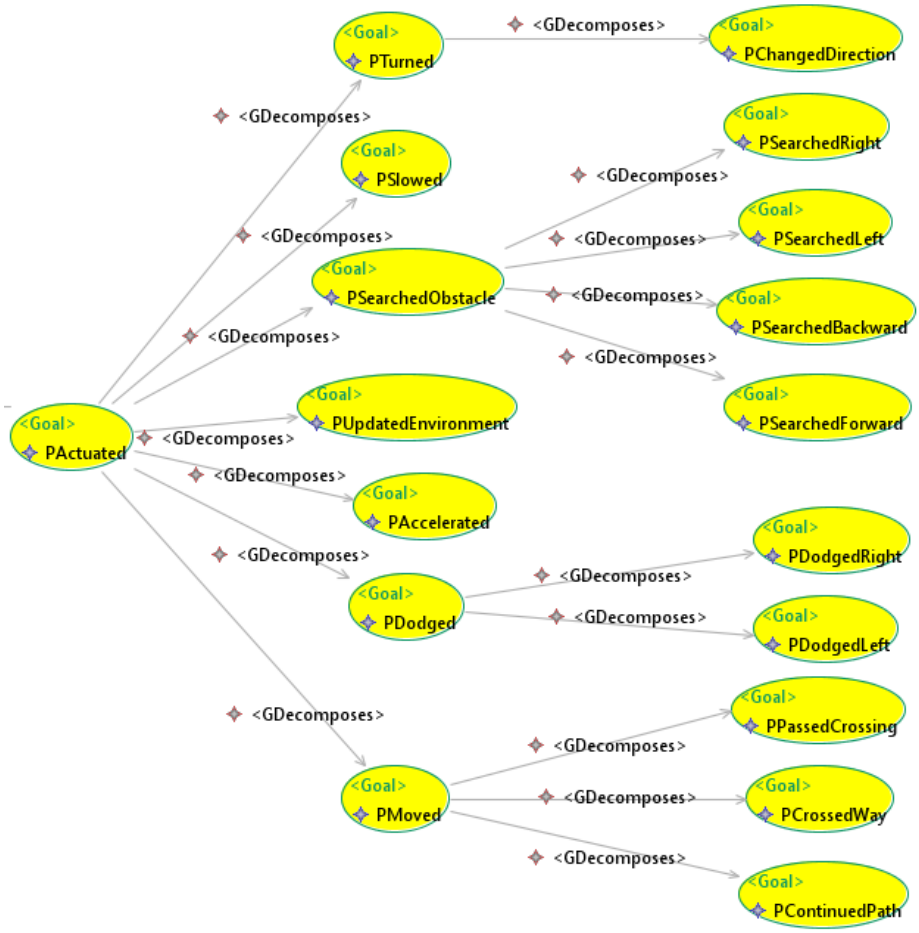

Figura 62: Composición OR de los objetivos de los peatones.

El nuevo objetivo principal introducido para los peatones queda descrito por la instancia de *Goal PArrivedFastDestination*. Esta instancia *Goal* tiene una descomposición *AND* formada por las instancias de *Goal PEndedRoute* y *PActuated* (ver Figura 61). Esta última instancia presenta una descomposición *OR* en siete instancias *Goal* que corresponden a acciones: *PMoved*, *PDodged*, *PAccelerated*, *PSearchedObstacle*, *PUpdatedEnvironment*, *PSlowed*,y *PTurned*. A su vez, *PMoved*

tiene una composición *OR* de tres instancias de *Goal*: *PContinuedPath*, *PCrossedWay* y *PPassedCrossing*, mientras que *PDodged* tiene una composición OR de dos instancias *Goal*: *GDodgedLeft* y *GDodgedRight*. *PSearchedObstacle* dispone de una composición *OR* formada por: *PSearchedForward*, *PSearchedBackward*, *PSearchedLeft* y *PSearchedRight*. Finalmente, *PTurned* se compone de la instancia *PChangedDirection* (ver Figura 62).

Como en la aproximación de la sección anterior, cada instancia de *Goal* tiene su propia instancia de *Task* encargada de proporcionar instrucciones cuando sea ejecutada. Estas instancias, al igual que las instancias de *Goal* asociadas, tienen modificado su nombre para diferenciarlas de las instancias de *Task* de la parte de especificación que considera a los conductores.

Respecto a las instancias de *Evaluator* encargadas de evaluar las distintas instancias *Goal* relacionadas con los peatones, se ha decidido en el planteamiento de modelado organizarlas mediante los niveles de composición de la estructura arbórea de dichas instancias. Se elige esta configuración para mostrar la flexibilidad de la propuesta y las diferentes formas de organizar los elementos de las especificaciones de modelos parciales con clúster *Interactivo*. Por lo tanto, esta especificación presenta cuatro instancias de *Evaluator* relacionadas de manera jerárquica. La instancia de *Goal* raíz *ArrivedFastDestination* es considerada sólo por la instancia de *Evaluator EvaluatorFirst*, mientras que las instancias de *Goal EndedRoute* y *Actuated* son comprobadas por la instancia *Evaluator EvaluatorSecond*. Las instancias de *Goal* de acciones (es decir, *PDodged* o *PSearchedObstacle*) son tenidas en cuenta por la instancia de *Evaluator EvaluatorThird*. Finalmente, las restantes instancias de *Goal* (es decir, *PContinuedPath* o *PSearchedLeft*) son consideradas por la instancia *EvaluatorFourth*.

Cuando se concluye la integración de las instancias de *Evaluator* con la estructura de instancias de *Goal*, se añade a la especificación de modelo una nueva instancia de *Actuator* llamada *PActuator*. Este *Actuator* ejecuta solamente las instrucciones de las instancias de *Task* integradas en la estructura arbórea de los objetivos de los peatones (cada instancia *Goal* debe tener asociada una instancia *Task*).

Con las modificaciones anteriores, en esta especificación de modelo conviven dos estructuras arbóreas, una para conductores y otra para peatones. Ambas cuentan con similares objetivos con composiciones *AND* y *OR*, pero con sus propias instancias de *Evaluator*, *Actuator* y *Task*.

6.3.3 Fase de generación preliminar de código

Una vez diseñadas las dos especificaciones de modelos, el siguiente paso consiste en realizar su transformación a código fuente. Esta tarea es llevada a cabo por el generador de código automáticamente al cargar los archivos XMI donde están almacenadas dichas especificaciones. A continuación se desarrollan y codifican las interacciones entre las instancias de las especificaciones.

Para el proyecto generado a partir de la especificación basada en factores de riesgo de los peatones jóvenes se reutiliza la fórmula de influencia basada en lógica borrosa del anterior caso de estudio. Para ello se insertan los fragmentos de código correspondientes en el cuerpo de los métodos *calculateXValue* y se almacena el estado actual.

El proyecto que contiene la especificación del clúster *Interactivo* reutiliza el código introducido en el anterior caso de estudio. Además, emplea como punto de partida el proyecto generado en esta misma fase del proceso para la nueva especificación. Por este motivo solamente hay que insertar fragmentos de código en las instancias relacionadas con los peatones. Así, se rellenan los métodos *evaluateGoals* y *executeChosenTask* de las nuevas instancias *Goal* y *Task* respectivamente. Finalmente el estado actual del proyecto es guardado.

6.3.4 Fase de especialización a la plataforma

Una vez concluidas las modificaciones en los proyectos, se han de integrar ambas especificaciones parciales para generar una especificación completa. En este caso, la especificación de modelo presenta dos instancias *Person* diferentes (*Driver* y *Pedestrian*). La instancia *Driver* se enlaza mediante la referencia *Pursues* con la instancia de *Goal ArrivedFastDestination*. Esto permite que los conductores dispongan de la estructura arbórea de objetivos creada en la sección anterior junto con las tareas asociadas. Por su parte, la instancia *Pedestrian* es combinada utilizando otra referencia *Pursues* con la instancia de *Goal PArrivedFastDestination*. A continuación, las instancias de *Person* se enlazan con sus instancias de *Evaluator* correspondientes mediante las referencias *Harness* e *IsHarnessed*. De esta manera, la instancia *Driver* queda unida a la instancia de *Evaluator EvaluateDestination*, mientras que la instancia *Pedestrian* queda conectada con la instancia *EvaluatorFirst*. Por último, las dos instancias de *Actuator*, *DActuator* y *PActuator* son enlazadas con las instancias *Driver* y *Pedestrian* respectivamente mediante referencias *Utilizes*.

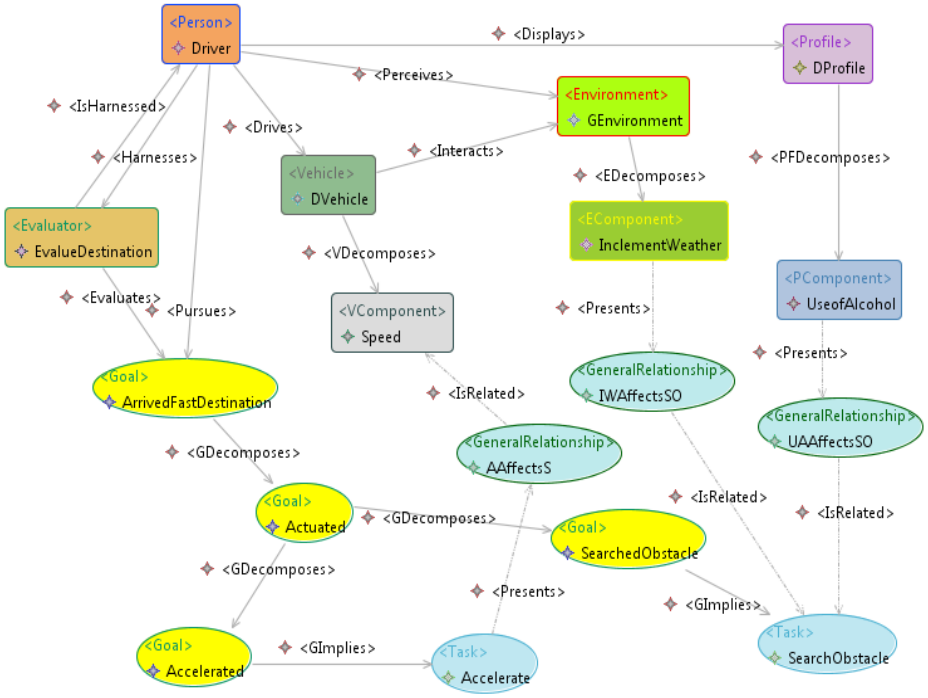

Figura 63: Extracto de la especificación completa para conductores.

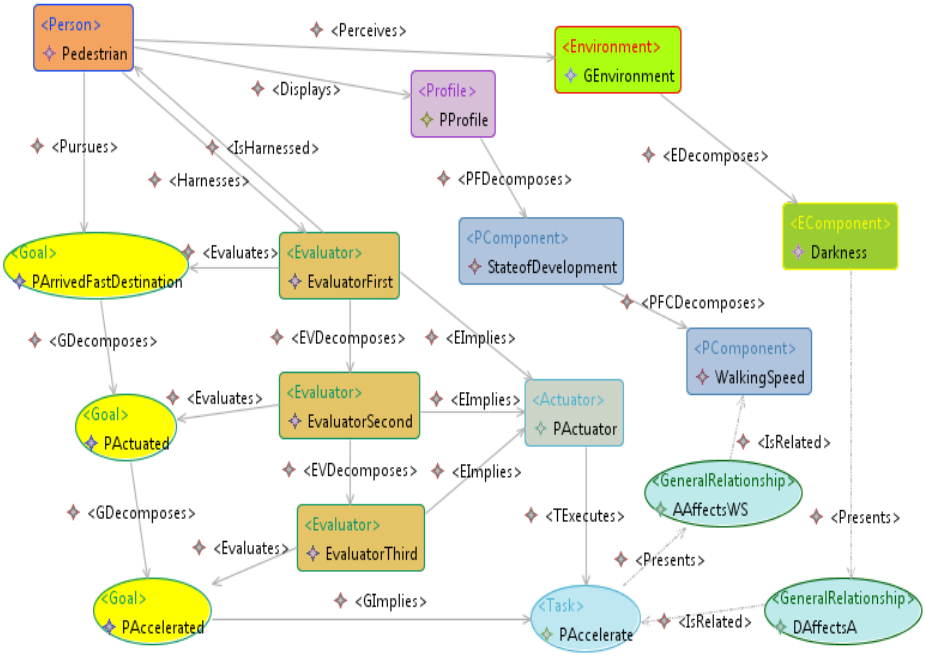

Figura 64: Extracto de la especificación completa para peatones.

A continuación se lleva a cabo la inserción de instancias de *GeneralRelationship* entre los elementos proporcionados por ambas especificaciones de modelos. Estas instancias se crean para marcar como los elementos de los clústeres *Mental* y *Entorno* afectan a las instancias de *Task*, y viceversa. Así, por parte de los conductores (ver Figura 63), instancias de *EComponent* como *InclementWeather* o *Darkness* afectan respectivamente a las instancias de *Task SearchObstacle* o *Accelerate*. También las instancias de *PComponent* (por ejemplo, *UseofAlcohol* o *DriverFatigue*) afectan a las instancias de *Task SearchObstacle*, mientras que las de *VComponent* (por ejemplo *Speed*) se ven afectadas por las instancias de *Task Accelerate* o *Brake*. En cuanto a los peatones (ver Figura 64), estos presentan instancias de *EComponent* (por ejemplo, *Darkness* o *TrafficDensity*) que afectan a las instancias de *Task Accelerate* y *Overtake*, e instancias de *PComponent* (por ejemplo, *Crowding* o *AttentionSpan*) que afectan a las instancias de *Task PAccelerate* y *PDodge*. También se consideran instancias de *Task* que influyen directamente en instancias de *PComponent*. Por ejemplo, la instancia de *Task PAccelerate* afecta directamente a la instancia de *PComponent WalkingSpeed*.

Después de haber introducido las *GeneralRelationship*, las instancias *Actuator* deben considerar mediante código los posibles valores de las instancias destino de las *GeneralRelationship* que presenten como origen instancias *Task* (por ejemplo *PAccelerate* puede modificar el atributo *PCValues* de la instancia *PComponent WalkingSpeed*). Para ello el método *executeChosenTask* debe de ser alterado para que esas actualizaciones en los valores de los atributos se produzcan cuando se ejecute la instancia *Task* correspondiente.

En este caso no es necesario agregar archivos externos mediante la herramienta generador de código, ya que el objetivo final consiste en generar un plug-in que actúe

como librería en la plataforma de test. Es decir, al contrario que en la sección anterior, será en la plataforma donde se creen los objetos pertenecientes a las clases de la especificación de modelo completa. En ese sentido, aquí no hay especialización a la plataforma destino. Indicar que en algunos casos pueden ser necesarios estos archivos externos si se desea crear objetos distintos a las clases de los elementos de la especificación de modelo. Un ejemplo de este tipo puede ser la implementación de casos de prueba unitarios, en los que haría falta la inserción de la librería JUnit [47].

Por último se genera el archivo de configuración siguiendo las pautas proporcionadas por el asistente correspondiente del generador de código. Después de esto se genera el plug-in final. Este archivo empaqueta el propio fichero de configuración, las clases generadas con los cambios introducidos en el código, la documentación asociada y el archivo XMI que contiene la especificación de modelo. Además, el archivo de manifiesto es modificado para que las clases generadas puedan ser consideradas por la plataforma. Estos pasos permiten la utilización del plug-in final como librería.

Al contrario que en la sección anterior, aquí debe ser la plataforma destino la adaptada a los requisitos del archivo resultante. Para ello, en la siguiente fase se deben crear nuevas clases que permitan considerar los artefactos proporcionados por esta fase.

6.3.5 Fase de simulación

Una vez completada la generación del plug-in se procede a cargarlo como librería en la plataforma de test. De esta manera pueden añadirse las nuevas funcionalidades que permiten a los agentes de la misma (conductor y peatón) basar sus decisiones en las teorías del dominio seleccionadas. El comportamiento por defecto de estos agentes se limita a seguir una ruta asignada previamente, sin realizar interacciones entre ellos o con el entorno que los rodea. Para poder utilizar la nueva funcionalidad hay que realizar por tanto cambios en el código de la plataforma.

Se ha de crear una nueva clase que se encargue de leer y almacenar los valores del archivo de configuración. También es necesaria una clase que cargue el contenido del archivo XMI proporcionado por el plug-in (ver Figura 65). Además, esta clase debe utilizar a la anterior para proporcionar los valores iniciales a cada elemento. Por su parte, las clases de los agentes deben ser modificadas para que consideren la especificación de modelo. Así, se crea un objeto de tipo *Driver* en los agentes conductor (ver Figura 66) y otro objeto de tipo *Pedestrian* en los peatones (ver Figura 67). Esto permite acceder a los elementos de la especificación, ya que la clase *Person* es la raíz de la misma. En cuanto a las clases *Task*, éstas deben ser extendidas para que tanto su atributo *Instructions* como su método *setInstructions* realicen las operaciones de acuerdo a las particularidades de la plataforma. Finalmente, se llevan a cabo cambios en la clase principal para poder realizar la instanciación de la clase encargada de leer y almacenar el fichero XMI de la especificación de modelo.

Ciertas modificaciones (por ejemplo, la carga del contenido de los archivos) necesitan cambios mínimos o pueden ser omitidas en ocasiones posteriores puesto que ya están incorporadas Estos cambios incrementales en la plataforma de simulación facilitan su especialización para soportar el LMT. En caso de no desear esta especialización se debe utilizar la opción de generación de adaptación a plataforma introducida en el caso de estudio anterior.

```
public Boolean generateInstanceModel() {

    Boolean correct = false;
    try {
        TrafficmodelPackage.eINSTANCE.eClass();
        Resource.Factory.Registry reg = Resource.Factory.Registry.INSTANCE;
        map = reg.getExtensionToFactoryMap();
        map.put("trafficmodel", new XMIResourceFactoryImpl());
        resSet = new ResourceSetImpl();
        resource = resSet.getResource(URI.createURI(trafficFile.toURI().toString()), true);
        trafficSim = (TrafficModelImpl) resource.getContents().get(0);
        correct = true;
    }
    catch (Exception e) {
        e.printStackTrace();
        correct = false;
    }
    return correct;
}
```

Figura 65: Carga del archivo XMI de diseño de la especificación.

```
public class DriverAgentDefault extends DriverAgent {

    public DriverAgentDefault() {

        super();
        DriverImpl driver = new DriverImpl();
    }
```

Figura 66: Modificaciones para incluir instancias *Driver*.

```
public class PedestrianAgentDefault extends PedestrianAgent {

    public PedestrianAgentDefault() {

        super();
        PedestrianImpl pedestrian = new PedestrianImpl();
    }
```

Figura 67: Modificaciones para incluir instancias *Pedestrian*.

6.4 Conclusiones

En este capítulo de tesis se han presentado dos ejemplos de utilización de la infraestructura de ISDM completa. El LMT se ha mostrado apropiado para realizar las especificaciones de modelos basadas en las teorías de tráfico seleccionadas. Por su parte, las herramientas de desarrollo (editor gráfico, generador de código y plataforma de test) han evidenciado su capacidad para asistir y facilitar el proceso de desarrollo.

Respecto al caso de estudio de la Sección 6.2, se ha realizado una especialización para la plataforma MATSim [90]. Para ello se ha partido de dos teorías de tráfico: una basada en un modelo de conducción llamado *Drivability* [5] y otra en un modelo de la toma de decisiones de los conductores [21]. Para la primera se ha realizado una especificación de modelo parcial utilizando instancias de las metaclases de los clústeres *Mental* y *Entorno* del LMT. En el caso de la segunda, se ha diseñado otra especificación de modelo parcial con instancias de las metaclases del clúster *Interactivo*. Ambas especificaciones han sido transformadas a código y modificadas. Después, se han integrado sus proyectos para obtener una especificación de modelo completa (es decir, con los tres clústeres del LMT). A continuación se ha añadido la plataforma MATSim como librería para especializar el código desarrollado a la misma. El resultado final ha sido la generación de una nueva plataforma con los elementos proporcionados por MATSim, junto a una serie de mejoras integradas. Estas mejoras están basadas en la inserción de elementos de soporte a las teorías.

En el caso de estudio de la Sección 6.3, se ha generado un plug-in integrable en la plataforma de test como librería. Para ello se ha utilizado una clasificación de tipos de individuos involucrados en el tráfico [78]. A partir de ella se ha generado una especificación de modelo parcial con instancias de los clústeres *Mental* y *Entorno*. Para la toma de decisiones se ha tomado el modelo del otro caso de estudio y extendido a los peatones acorde con la teoría en [25]. Al igual que en el caso anterior, se ha procedido a combinar las dos especificaciones en un único modelo. Aquí se han reutilizando otros elementos del anterior caso de estudio, como la fórmula de influencia entre componentes y su codificación. Por último, se ha llevado a cabo la generación del plug-in y se han introducido modificaciones en la plataforma de test para adaptarla a las necesidades requeridas por éste.

El análisis de los diferentes casos de estudio permite extraer conclusiones acerca de las ventajas y desventajas en el uso de la propuesta. Las principales se resumen a continuación.

Buena parte de la especificación de la simulación se realiza en base a modelos con el LMT. Ello reduce el papel del código y facilita una mayor implicación de los expertos en tráfico.

La reutilización en general es muy alta cuando existen características compartidas entre proyectos. Por ejemplo, se puede reutilizar una misma plataforma destino con una teoría de interacción de los individuos completamente desarrollada y contexto de tráfico diferentes (clústeres *Mental* y *Entorno*). Los cambios requeridos para ello se producen principalmente en las especificaciones, siendo menores en las partes del código vinculadas a los elementos de los modelos.

El soporte a las tareas mediante herramientas reduce también la carga de trabajo, y facilita centrar el esfuerzo en los modelos, pudiendo reutilizar los proyectos anteriores. Por ejemplo, el generador de código realiza integraciones de modelos enlazando elementos de sus clústeres, y permite crear nuevas clases o extensiones de otras ya existentes, lo que permite ahorrar esfuerzo en la codificación inicial de las mismas.

Quizás la herramienta de soporte más destacable de la propuesta, dado su peso en el proceso de desarrollo, es el generador de código. Su organización basada en asistentes le permite proporcionar una funcionalidad extensible tanto durante la generación de especificaciones de modelos como al trabajar con fragmentos de código fuente. Entre sus funcionalidades, el empaquetado del estado del proyecto ofrece una manera sencilla de reutilizar los artefactos generados.

La principal desventaja de la propuesta consiste en el esfuerzo a realizar para desarrollar el marco de desarrollo. También el uso de un proceso de desarrollo que no es el habitual, especialmente para los programadores. Sin embargo, una vez hecho esto, este esfuerzo es altamente compensado por la capacidad de reutilización de artefactos que permite, los diseños visuales, y la simplificación de las tareas más complejas mediante asistentes y automatismos.

Como conclusión, indicar que el marco de desarrollo de ISDM propuesto ha mostrado su viabilidad para cumplir con los objetivos propuestos. Los casos de estudio han sido capaces de generar simulaciones que integran diferentes teorías de tráfico, toma de decisiones e interacción. Además, éstas han podido ser adaptadas con un esfuerzo reducido a diferentes plataformas de simulación. La base del LMT en el modelado basado en agentes y el modelo CVE, junto con los mecanismos de extensión y especialización, han mostrado ser apropiados para modelar diferentes aspectos de tráfico e integrar múltiples modelos teóricos.

Capítulo 7. Conclusiones finales

Este capítulo final recoge las principales aportaciones de esta tesis, e introduce brevemente las líneas futuras de investigación.

7.1 Aportaciones

Esta tesis ha presentado un marco de desarrollo de ISDM especializado en simulaciones de tráfico. Desde el punto de vista conceptual, se centra en el comportamiento de los individuos y las interacciones entre estos y con su entorno. Dada la variedad de necesidades, teorías e infraestructuras diferentes que se emplean en el desarrollo de estas simulaciones, el marco ha sido diseñado prestando una especial atención a facilitar su modificación y la integración de nuevos elementos. El otro objetivo principal ha sido dotar de una mayor autonomía en el desarrollo de las simulaciones a los expertos en tráfico, de manera que puedan participar de una forma más activa en el mismo. El marco está constituido por el LMT, tres herramientas de desarrollo específicas de este trabajo, y un proceso de desarrollo con los anteriores elementos.

El LMT se centra en las personas participantes en el tráfico. Adopta una aproximación basada en agentes [2, 35] y extiende el modelo CVE [1] para organizar sus conceptos. Extendiendo este modelo considera que las personas pueden jugar tres roles fundamentales en el tráfico: conductores, peatones y pasajeros. Estos roles determinan sus potenciales relaciones con el entorno.

La organización de los elementos del LMT se estructura en torno a la especificación de diferentes aspectos de los problemas de tráfico. Esto da lugar a tres grupos de elementos denominados clústeres: *Mental*, *Entorno* e *Interactivo*. El primero se fundamenta en las teorías de [79] y representa las características propias y el conocimiento actual de un individuo o conjunto de ellos. El segundo se basa en el modelo CVE. Este modelo considera que los conductores interactúan con sus vehículos y el resto del entorno para obtener información que guíe su actuación. Este clúster extiende este modelo para considerar los otros dos tipos de personas que participan en el tráfico (peatones y pasajeros). Finalmente, el clúster *Interactivo* adapta elementos y conceptos procedentes del paradigma de agentes [93], especialmente nociones relacionadas con la toma de decisiones del modelo CDI [69] y metodologías como Tropos [9] o INGENIAS [64]. Este clúster modela un ciclo de percepción, razonamiento y actuación en base a objetivos que pueden ser alcanzados a través de tareas.

El LMT también busca que su marco conceptual sea flexible y extensible, con el fin de facilitar la integración de diferentes teorías y aspectos de la simulación. Para ello,

incluye mecanismos para soportar jerarquías de herencia y composición. La herencia permite realizar especializaciones de los conceptos del lenguaje (tanto entidades como relaciones), y puede ser múltiple. La composición permite establecer las partes de un elemento dado, soportando niveles de anidamiento.

Para la implementación del LMT y las herramientas de desarrollo relacionadas se han utilizado los proyectos de Eclipse para ISDM, como EMF [85] y GEF [73]. El LMT se define por ello con un metamodelo especificado con el lenguaje de metamodelado Ecore. En base a este metamodelo se han desarrollado tres herramientas específicas: un editor de modelos, un generador de código y una plataforma de simulación de test.

El editor gráfico permite la especificación visual de modelos conformes al LMT. Su implementación es una utilización directa de GEF.

El generador de código tiene como misión realizar las transformaciones a código fuente de las especificaciones de modelos proporcionadas por el editor. Para ello permite navegar y completar las especificaciones, y vincularles código. Para completar las especificaciones, la herramienta facilita la integración de especificaciones de modelos parciales a través de nuevas instancias de relaciones que enlacen sus elementos. A los elementos de las especificaciones se les vincula código para guiar la generación. El código se puede incorporar a elementos ya existentes o bien definir nuevos. Para ello se parte de las plantillas de código que EMF genera automáticamente para el metamodelo del LMT. También se pueden insertar librerías externas, por ejemplo para describir la plataforma de simulación de tráfico destino y permitir utilizar clases específicas de la misma.

La plataforma de test permite realizar simulaciones microscópicas de tráfico. Considera solamente peatones y conductores. Su implementación se basa en la plataforma de agentes A-Globe [82] y los mapas del SDK de Esri [51]. Obtiene el modelo de entorno de los datos proporcionados por los servicios web de OpenStreetMap [28]. La plataforma se complementa con una aplicación servidor encargada de dar soporte en la generación de rutas. Esto permite aliviar la carga computacional de ejecutar la plataforma de simulación, al delegar parte de los cálculos al servidor.

En lo referente al proceso de desarrollo, este es un proceso iterativo e incremental compuesto de cinco fases principales: *Evaluación preliminar de la teoría*, *Diseño de la especificación de modelo*, *Generación preliminar de código*, *Especialización a la plataforma*, y *Simulación*. La primera fase evalúa la teoría de tráfico seleccionada y realiza un planteamiento de modelado preliminar de la futura especificación de modelo. La segunda desarrolla las especificaciones según el planteamiento de la fase anterior. La tercera transforma a código fuente las especificaciones. Este código fuente se modifica para completar los cuerpos de los métodos predefinidos, o añadir atributos y métodos no existentes si es necesario. Además, se desarrollan y codifican los mecanismos de influencia a través de las distintas relaciones entre las instancias de los elementos de los clústeres *Mental* y *Entorno*. La cuarta fase se ocupa de modificar el código fuente para su especialización a la plataforma destino. Además, se particulariza la toma de decisiones de los individuos introduciendo cambios en el código fuente de las instancias de los elementos del clúster *Interactivo*. El resultado de esta fase es un archivo comprimido. Este archivo puede ser un plug-in utilizable como librería en la plataforma destino o un fichero que contenga la plataforma de simulación destino junto a las modificaciones realizadas en ella para que soporte los elementos de la

especificación de modelo correspondiente. Finalmente, la quinta fase configura la plataforma destino si fuese necesario y ejecuta la simulación resultante.

Los participantes en este proceso pueden jugar dos roles: el *experto en tráfico* y el *programador*. El primero soporta la mayor carga de trabajo y está presente en todas las fases del proceso, mientras que el segundo colabora sólo en momentos puntuales, especialmente de las tres últimas fases.

La evaluación en los casos de estudio apoya la adecuación del marco de desarrollo propuesto para satisfacer los objetivos de esta tesis. La mayor parte de la especificación a alto nivel de las simulaciones se puede hacer mediante especificaciones de modelos con el LMT. Estas especificaciones pueden ser desarrolladas por los expertos en tráfico, ya que se trata de un lenguaje específico de dominio para su campo. Además, el hecho de que se trate de unas especificaciones a un mayor nivel de abstracción que el código favorece su reutilización entre proyectos. Los siguientes pasos se basan en la asignación de fragmentos de código para especificar el comportamiento de los elementos de las especificaciones, principalmente en sus métodos. Estos fragmentos pueden estar especializados según teorías y plataformas, lo que favorece su reutilización. Su desarrollo inicial es obra de los programadores, pero una vez creados y probados pueden ser asociados por los propios expertos en tráfico a los elementos de sus modelos usando las herramientas. De esta manera, los expertos pueden generar de forma autónoma buena parte de la simulación. Con esta organización del desarrollo, la mayor parte de las modificaciones de una simulación consisten en cambios en los modelos y la asignación de nuevos fragmentos de código.

Esta organización del desarrollo satisface los objetivos iniciales: dota de una mayor relevancia en el proceso a los expertos en tráfico; reduce el coste del desarrollo; hace explícita toda la información usada en la generación de la simulación de tráfico final; y favorece la trazabilidad entre los distintos artefactos involucrados en el desarrollo, lo que mejora su validación.

7.2 Trabajo futuro

El marco de desarrollo presentado en esta tesis es aún trabajo abierto, pendiente de evolución en varios frentes. Aunque los casos de estudio han validado la propuesta probando su utilidad, también han señalado potenciales aspectos de mejora.

El LMT, siendo el núcleo del marco de desarrollo, es el elemento que debe ser modificado para abordar nuevas necesidades. En la actualidad se centra en simulaciones microscópicas del tráfico basadas en agentes, con solo aspectos limitados aplicables a los grupos de agentes y que lo conectan con las simulaciones mesoscópicas. Como muestra el trabajo relacionado en el área, las simulaciones que involucran muchos elementos o con un alto nivel de detalle suelen requerir enfoques mesoscópicos. Aspectos como la definición de características o comportamiento de grupos, sistemas normativos o flujos de tráfico, han de ser considerados en el LMT. Otro aspecto no contemplado en la actualidad es la dimensión temporal. Por ejemplo, los momentos en los que deben ocurrir ciertos eventos. Ello requiere incorporar al LMT diagramas que reflejen el comportamiento dinámico del sistema además del estructural disponible actualmente.

El metamodelo también ofrece oportunidades de rediseño según el uso observado, para simplificar partes de su estructura y evitar redundancias. Por ejemplo, se plantea introducir una metaclase abstracta intermedia llamada *BehavioralElement* que herede de *GeneralElement*. Las metaclases de los clústeres *Mental* y *Entorno* heredarían de esta nueva metaclase. En la metaclase *GeneralElement* se introduciría un atributo llamado *Name* que evitaría tener que definir en cada una de las metaclases de los tres clústeres su propio atributo para el nombre. En cuanto a *BehavioralElement*, dispondría de un atributo *Value* y un método que permitiese calcularlo llamado *calculateValue*. De esta forma se evitaría que cada una de las clases de los clústeres *Mental* y *Entorno* tuvieran que definir los suyos propios.

Las restricciones son otro aspecto de especificación, tanto del LMT como de las especificaciones de modelos, que está pendiente de mejora. En la actualidad se realiza en base a restricciones OCL [57], que requieren un conocimiento experto de este lenguaje y de metamodelos. Su uso resulta complejo para los expertos en tráfico, por lo que sería recomendable disponer de primitivas en el LMT para poder incorporar en las especificaciones al menos algunos tipos frecuentes de restricciones. Otra posibilidad sería hacerlo a través de asistentes especializados en el editor de modelos y en el generador de código.

Las herramientas constituyen otro punto donde existen aspectos pendientes. El principal es la adaptación de la infraestructura de desarrollo a cambios. Pese a las facilidades de los proyecto Eclipse, modificar las herramientas cuando se produce un cambio en el LMT requiere mucho trabajo manual. Generalmente es necesario rescribir buena parte de los ficheros de especificación de las mismas. Aquí sería deseable disponer de meta-editores que facilitaran estas tareas de edición del metamodelo y regeneración de las herramientas. Existen ya trabajos en esta línea en la ISDM y la ISOA, como el marco de desarrollo de aplicaciones de INGENME [65].

El otro punto que puede ofrecer una mayor ganancia en el proceso de desarrollo de simulaciones es la integración del editor de modelos y el generador de código. En la actualidad se trata de dos herramientas separadas, pero el trabajo de los expertos sugiere que sería más apropiado poderlas usar conjuntamente en los flujos de trabajo. Típicamente se requiere combinar la especificación de modelos con la integración de modelos parciales y la vinculación a fragmentos de código. También se persigue poder generar de forma rápida pruebas sobre el estado actual de los artefactos del desarrollo. En esta línea ya hay una primera aproximación con el asistente de integración de modelos en el generador de código, que permite una edición limitada de los modelos restringida al clúster *Interactivo*. El asistente de integración podría ser adaptado para permitir diseñar especificaciones de modelos completas que tuviesen instancias de elementos de los tres clústeres. De esta forma, el editor gráfico podría ser sustituido por este asistente, centralizando el proceso de desarrollo en el generador de código y simplificando la infraestructura.

La plataforma de test debe mejorar principalmente el cálculo de rutas que utiliza, ya que consume gran cantidad de recursos computacionales. Aunque esto ha sido aliviado mediante la aplicación servidor, este proceso sigue siendo costoso. Además, se debe optimizar el comportamiento básico de los agentes peatones (por ejemplo, aplicando alguno de los modelos vistos en la Sección 2.2.2).

Los cambios en los elementos anteriores producirán con toda probabilidad modificaciones en la definición del proceso de desarrollo. Por ejemplo, se deberán dar

pautas para identificar nuevos conceptos en el LMT o utilizar nuevas herramientas y funcionalidades.

Otro aspecto pendiente es una evaluación más amplia de la propuesta. Aunque se han realizado casos de estudio variados en cuanto a contextos de tráfico considerados, teorías y plataformas, estos no bastan para afirmar que la propuesta del marco de desarrollo es capaz de soportar un espectro amplio de simulaciones de tráfico. Por ejemplo, no hay pruebas con entornos sometidos a múltiples señales de tráfico, sino solo con semáforos y límites de velocidad. Tampoco hay pruebas con entornos distribuidos de simulación donde estos aspectos tengan que ser considerados de algún modo en la especificación o la especialización a la plataforma. Por tanto, es necesario experimentación adicional para comprobar si la propuesta aquí desarrollada produce simulaciones apropiadas independientemente de cuál sea el contexto, teoría y plataforma destino. Además, se deben realizar pruebas orientadas a comprobar si el marco de desarrollo es adecuado para satisfacer diferentes necesidades de análisis, por ejemplo, para estudiar la planificación urbana, los cambios normativos, o características de los vehículos.

Finalmente, sería interesante poder inyectar datos reales sobre la dinámica del tráfico en las simulaciones. Esto podría añadir información sobre las maniobras habituales de conducción en determinados lugares, los atascos de tráfico o la programación de los semáforos.

Bibliografía

1. Amditis, A., Pagle, K., Joshi, S., y Bekiaris, E. (2010). Driver-Vehicle-Environment monitoring for on-board driver support systems: Lessons learned from design and implementation. *Applied Ergonomics*, 41(2), 225-235. Elsevier.

2. Axtell, R. L. y Epstein, J. M. (1994). Agent-based modeling: understanding our creations. *The Bulletin of the Santa Fe Institute*, 9(2), 28-32.

3. Bauer, D., Seer, S., y Brändle, N. (2007). Macroscopic pedestrian flow simulation for designing crowd control measures in public transport after special events. En *2007 Summer Computer Simulation Conference (SCSC 2007)*, 1035-1042. Society for Computer Simulation International.

4. Behrisch, M., Bieker, L., Erdmann, J., y Krajzewicz, D. (2011). SUMO–Simulation of Urban MObility. En *3rd International Conference on Advances in System Simulation (SIMUL 2011)*, 63–68. IARIA.

5. Bekiaris, E., Amditis, A., y Panou, M. (2003). DRIVABILITY: a new concept for modelling driving performance. *Cognition, Technology & Work*, 5(2), 152-161. Springer-Verlag.

6. Bellifemine, F., Poggi, A., y Rimassa, G. (1999). JADE–A FIPA-compliant agent framework. En *4th Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM 1999)*, 97-108.

7. Ben-Akiva, M., De Palma, A., y Isam, K. (1991). Dynamic network models and driver information systems. *Transportation Research Part A: General*, 25(5), 251-266. Elsevier.

8. Bézivin, J. (2004). In search of a basic principle for model driven engineering. *Novatica Journal, Special Issue*, 5(2), 21-24.

9. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., y Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3), 203-236. Springer-Verlag.

10. Black, A. (1995). *Urban Mass Transportation Planning*. McGraw Hill.

11. Bone, S. A., y Mowen, J. C. (2006). Identifying the traits of aggressive and distracted drivers: A hierarchical trait model approach. *Journal of Consumer Behaviour*, 5(5), 454-464. John Wiley & Sons.

12. Burrough, P. A. (1986). *Principles of Geographical Information Systems for Land Resources Assessment.* Oxford University Press.

13. Chang, L. Y., y Chien, J. T. (2013). Analysis of driver injury severity in truck-involved accidents using a non-parametric classification tree model. *Safety Science*, 51(1), 17-22. Elsevier.

14. Czarnecki, K., y Helsen, S. (2006). Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3), 621-645.IBM.

15. Deci, E. L., y Ryan, R. M. (1991). A motivational approach to self: Integration in personality. En *Nebraska Symposium on Motivation: Perspectives on Motivation*, 38, 237-288. University of Nebraska Press.

16. Dia, H. (2002). An agent-based approach to modelling driver route choice behaviour under the influence of real-time information. *Transportation Research Part C: Emerging Technologies*, 10(5), 331-349. Elsevier.

17. Doherty, S. T., Andrey, J. C., y MacGregor, C. (1998). The situational risks of young drivers: The influence of passengers, time of day and day of week on accident rates. *Accident Analysis & Prevention*, 30(1), 45-52. Elsevier.

18. Ehlert, P. A., y Rothkrantz, L. J. (2001). A reactive driving agent for microscopic traffic simulation. En *15th European Simulation Multiconference (ESM 2001)*, 943-949.

19. Esser, J., y Schreckenberg, M. (1997). Microscopic simulation of urban traffic based on cellular automata. *International Journal of Modern Physics C*, 8(5), 1025-1036. World Scientific Publishing.

20. Fastenmeier, W., y Gstalter, H. (2007). Driving task analysis as a tool in traffic safety research and practice. *Safety Science*, 45(9), 952-979. Elsevier.

21. Fernández-Isabel, A., y Fuentes-Fernández, R. (2011). An Agent-Based Platform for Traffic Simulation. En *6th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2011)*, Advances in Intelligent and Soft Computing 87, 505-514. Springer-Verlag.

22. France, R., y Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. En *2007 Future of Software Engineering (FOSE 2007)*, 37-54. IEEE.

23. Fuentes-Fernández, R., Hassan, S., Pavón, J., Galán, J. M., y López-Paredes, A. (2012). Metamodels for role-driven agent-based modelling. *Computational and Mathematical Organization Theory*, 18(1), 91-112. Springer-Verlag.

24. Fuller, R. (1984). A conceptualization of driving behaviour as threat avoidance. *Ergonomics*, 27(11), 1139-1155, Taylor & Francis.

25. Gipps, P. G., y Marksjö, B. (1985). A micro-simulation model for pedestrian flows. *Mathematics and Computers in Simulation*, 27(2), 95-105. Elsevier.

26. Greenberg, H. (1959). An analysis of traffic flow. *Operations Research*, 7(1), 79-85. INFORMS.

27. Gronback, R. C., y Merks, E. (2008). Model Driven Architecture at Eclipse. *Upgrade*, 9(2), 35-39.

28. Haklay, M., y Weber, P. (2008). OpenStreetMap: User-generated street maps. *Pervasive Computing*, 7(4), 12-18. IEEE.

29. Haque, M. M., Chin, H. C., y Huang, H. (2009). Modeling fault among motorcyclists involved in crashes. *Accident Analysis & Prevention*, 41(2), 327-335. Elsevier.

30. Helbing, D., y Molnar, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, 51(5), 4282-4286. American Physical Society.

31. Hoogendoorn, S., y Bovy, P. (2002). Normative pedestrian behaviour theory and modelling. *Transportation and Traffic Theory*, 15, 219-245.Elsevier.

32. Hoogendoorn, S. P. (2003). Microscopic simulation of pedestrian flows. En *82nd Annual Meeting at the Transportation Research Board (TRB 2003)*, 1-11.

33. Hoogendoorn, S. P., y Daamen, W. (2004). Design assessment of Lisbon transfer stations using microscopic pedestrian simulation. En *Computers in railways IX (Congress Proceedings of CompRail 2004)*, 135-147. WIT Press.

34. Horni, A., y Axhausen, K. W. (2012). *MATSim Agent Heterogeneity and a One-Week Scenario*. Technical Report, Institute for Transport Planning and Systems & Swiss Federal Institute of Technology Zurich.

35. Janssen, M. A. (2005). Agent-based modelling. *Modelling in Ecological Economics*, 155-172. Edward Elgar Publishing.

36. Kamaruddin, N., y Wahab, A. (2011). Heterogeneous driver behavior state recognition using speech signal. En *10th WSEAS International Conference on System Science and Simulation in Engineering (ICOSSSE 2011)*, 207-212.

37. Kashani, H. R., y Saridis, G. N. (1983). Intelligent control for urban traffic systems. *Automatica*, 19(2), 191-197. Elsevier.

38. Kent, S. (2002). Model driven engineering. En *Integrated Formal Methods*, Lecture Notes in Computer Science, 2335, 286-298. Springer-Verlag.

39. Kidd, E. A., y Laughery, K. R. (1964). *A computer model of driving behavior: The highway intersection situation.* CAL REPORT, NVJ (1843).

40. Klir, G., y Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic*. New Jersey: Prentice Hall.

41. de Lara, J., Vangheluwe, H., y Mosterman, P. J. (2006). Modelling and analysis of traffic networks based on graph transformation. En *5th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems (FORMS/FORMATS 2004)*, 120-127. Technical University of Braunschweig, Institute for Traffic Safety and Automation Engineering.

42. Liebner, M., Baumann, M., Klanner, F., y Stiller, C. (2012). Driver intent inference at urban intersections using the intelligent driver model. En *Intelligent Vehicles Symposium (IV 2012)*, 1162-1167. IEEE.

43. Lind, J. (2001). Issues in agent-oriented software engineering. En *Agent-Oriented Software Engineering*, Lecture Notes in Computer Science, 1957, 45-58. Springer-Verlag.

44. MacAdam, C. C. (1981). Application of an optimal preview control for simulation of closed-loop automobile driving. En *IEEE Transactions on Systems, Man and Cybernetics*, 11(6), 393–399, IEEE.

45. Maciejewski, M., y Nagel, K. (2011). Towards multi-agent simulation of the dynamic vehicle routing problem in MATSim. En *9th Parallel Processing and Applied Mathematics (PPAM 2011*), Lecture Notes in Computer Science, 7204, 551-560. Springer Berlin Heidelberg.

46. Malta, L., Miyajima, C., y Takeda, K. (2009). A study of driver behavior under potential threats in vehicle traffic. *IEEE Transactions on Intelligent Transportation Systems*, 10(2), 201-210, IEEE.

47. Massol, V., y Husted, T. (2003). *Junit in Action*. Manning Publications Co.

48. Matsushita, S., y Okazaki, S. (1991). A study of simulation model for way finding behavior by experiments in mazes. *Journal for Architecture, Planning, and Environmental Engineering*, 429, 51. Japan Publications Trading Co Ltd.

49. May, A. D. (1990). *Traffic Flow Fundamentals*. Prentice Hall.

50. Messner, A., y Papageorgiou, M. (1990). METANET: A macroscopic simulation program for motorway networks. *Traffic Engineering & Control*, 31(8-9), 466-470. The Hemming Group Ltd.

51. Mitchell, A. (1999). The *ESRI Guide to GIS Analysis: Geographic Patterns & Relationships*, 1. ESRI, Inc.

52. Näätänen, R., y Summala, H. (1974). A model for the role of motivational factors in drivers' decision-making. *Accident Analysis & Prevention*, 6(3), 243-261. Elsevier.

53. Ni, D. (2006). A framework for new generation transportation simulation. En *38th Winter Simulation Conference (WSC 2006)*, 1508-1514. Winter Simulation Conference.

54. Neighbors, C., Vietor, N. A., y Knee, C. R. (2002). A motivational model of driving anger and aggression. *Personality and Social Psychology Bulletin*, 28(3), 324-335. SAGE Publications Inc.

55. Nielsen, T. D., y Jensen, F. V. (2009). *Bayesian Networks and Decision Graphs*. Springer-Verlag.

56. Object Management Group: Software & Systems Process Engineering Meta-Model Specification, v2.0. (2008). *http://www.omg.org/spec/SPEM/* [Acceso Online: 12-Octubre-2015]

57. Object Management Group: Object Constraint Language Specification, v2.4. (2014). *http://www.omg.org/spec/OCL/ [*Acceso Online: 12-Octubre-2015]

58. Object Management Group: OMG Meta-Object Facility (MOF) Core Specification, v2.5 (2015). *http://www.omg.org/spec/MOF/2.5/* [Acceso Online: 12-Octubre-2015]

59. Object Management Group: OMG Unified Modeling Language (OMG UML), v2.5 (2015). *http://www.omg.org/spec/UML/2.5/* [Acceso Online: 12-Octubre-2015]

60. Object Management Group: XML Metadata Interchange Specification, v2.5.1 (2015). *http://www.omg.org/spec/XMI/* [Acceso Online: 12-Octubre-2015]

61. Oren, T. I., y Ghasem-Aghaee, N. (2003). Personality representation processable in fuzzy logic for human behavior simulation. En *Summer Computer Simulation Conference (SCSC 2003)*, 11-18. Society for Computer Simulation International.

62. Paruchuri, P., Pullalarevu, A. R., y Karlapalem, K. (2002). Multi agent simulation of unorganized traffic. En *1st International Joint Conference on Autonomous Agents and Multiagent Systems: part 1 (AAMAS 2002)*, 176-183. ACM.

63. Pavón, J., y Gómez-Sanz, J. (2003). Agent oriented software engineering with INGENIAS. En *Multi-Agent Systems and Applications III*, Lecture Notes in Computer Science, 2691, 394-403. Springer-Verlag.

64. Pavón, J., Gómez-Sanz, J. J., y Fuentes, R. (2005). The INGENIAS methodology and tools. En *Agent-Oriented Methodologies*, capítulo 9, 236-276. IGI Publishing.

65. Pavón, J., Gómez-Sanz, J., y Paredes-López, A. (2011). The SiCoSSyS approach to SoS engineering. En *6th International Conference on System of Systems Engineering (SoSE 2011)*, 179-184. IEEE.

66. Predtechenskii, V. M., y Milinskiĭ, A. I. (1978). *Planning for Foot Traffic Flow in Buildin*gs. National Bureau of Standards, US Department of Commerce, and the National Science Foundation, Washington, DC.

67. Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257-286. IEEE.

68. Ranney, T. A. (1999). Psychological factors that influence car-following and car-following model development. *Transportation Research Part F: Traffic Psychology and Behaviour*, 2(4), 213-219. Elsevier.

69. Rao, A. S., y Georgeff, M. P. (1991). Modeling rational agents within a BDI architecture. En *2nd International Conference on Principles of Knowledge Representation and Reasoning (KR 1991)*, 473-484. Morgan Kaufmann.

70. Rasmussen, C. E. y Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

71. Rebollar, A. M., Esquivel, H. E., y Moreno, L. A. G. (2009). Una guía rápida de la metodología TROPOS. *Gerencia Tecnológica Informática*, 7, 67-77.

72. Rosenbloom, T. (2009). Crossing at a red light: Behaviour of individuals and groups. *Transportation Research Part F: Traffic Psychology and Behaviour*, 12(5), 389-394. Elsevier.

73. Rubel, D., Wren, J., y Clayberg, E. (2011). *The Eclipse Graphical Editing Framework (GEF)*. Addison-Wesley Professional.

74. Rubinstein, R. Y., y Kroese, D. P. (2011). *Simulation and the Monte Carlo Method*, 2nd ed. John Wiley & Sons.

75. Salvucci, D. D., y Gray, R. (2004). A two-point visual control model of steering. *Perception*, 33(10), 1233-1248. SAGE Publications Inc.

76. Sasaki, M., y Nagatani, T. (2003). Transition and saturation of traffic flow controlled by traffic lights. *Physica A: Statistical Mechanics and its Applications*, 325(3), 531-546. Elsevier.

77. Sathyanarayana, A., Boyraz, P., y Hansen, J. H. (2008). Driver behavior analysis and route recognition by hidden Markov models. En *2008 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2008)*, 276-281. IEEE.

78. Schieber, R. A., y Thompson, N. J. (1996). Developmental risk factors for childhood pedestrian injuries. *Injury Prevention*, 2(3), 228-236. BMJ Publishing Group Ltd.

79. Shinar, D. (1978). *Psychology on the Road. The Human Factor in Traffic Safety*. John Wiley and Sons.

80. Shinar, D., Meir, M., y Ben-Shoham, I. (1998). How automatic is manual gear shifting? *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 40(4), 647-654. SAGE Publications Inc.

81. Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 60(1), 51-92. Elsevier.

82. Sislak, D., Rehak, M., y Pechoucek, M. (2005). A-Globe: Multi-Agent Platform with Advanced Simulation and Visualization Support. En *2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2005)*, 805-808. IEEE.

83. Sprinkle, J., Rumpe, B., Vangheluwe, H., y Karsai, G. (2010). Metamodelling. En *Model-Based Engineering of Embedded Real-Time Systems*, Lecture Notes in Computer Science, 6100, 57-76. Springer-Verlag.

84. Stanton, N. A., y Salmon, P. M. (2009). Human error taxonomies applied to driving: A generic driver error taxonomy and its implications for intelligent transport systems. *Safety Science*, 47(2), 227-237. Elsevier.

85. Steinberg, D., Budinsky, F., Merks, E., y Paternostro, M. (2008). *EMF: Eclipse Modeling Framework*. Pearson Education.

86. Stober, T., y Hansmann, U. (2010). Traditional Software Development. En *Agile Software Development*, 15-33. Springer Berlin Heidelberg.

87. Tezuka, S., Soma, H., y Tanifuji, K. (2006). A study of driver behavior inference model at time of lane change using Bayesian networks. En *IEEE International Conference on Industrial Technology (ICIT 2006)*, 2308-2313. IEEE.

88. Thompson, P. A., y Marchant, E. W. (1995). A computer model for the evacuation of large building populations. *Fire Safety Journal*, 24(2), 131-148. Elsevier.

89. Tolujew, J., y Alcalá, F. (2004). A mesoscopic approach to modeling and simulation of pedestrian traffic flows. En *18th European Simulation Multi-Conference, (ESM 2004)*, 13-16.

90. Transport Systems Planning and Transport Telematics Group, Transport Planning Group and Senozon Company. MATSim, Multi-agent transportation simulation, *www.matsim.org* [Acceso Online: 12-Octubre-2015].

91. Van Aerde, M., Hellinga, B., Baker, M., y Rakha, H. (1996). INTEGRATION: An overview of traffic simulation features. En 1996 *Transportation Research Board Annual Meeting*, 1-14.

92. Van den Berg, M., Hegyi, A., De Schutter, B., y Hellendoorn, J. (2003). A macroscopic traffic flow model for integrated control of freeway and urban traffic networks. En *42nd IEEE Conference on Decision and Control (CDC 2003)*, 3, 2774-2779. IEEE.

93. Van der Hoek, W., y Wooldridge, M. (2008). Multi-agent systems. *Foundations of Artificial Intelligence*, 3, 887-928. Elsevier.

94. Wahab, A., Wen, T. G., y Kamaruddin, N. (2007). Understanding driver behavior using multi-dimensional CMAC. En *6th International Conference on Information, Communications & Signal Processing (ICICS 2007)*, 1-5. IEEE.

95. Wilde, G. J. (1982). The theory of risk homeostasis: implications for safety and health. *Risk Analysis*, 2(4), 209-225. John Wiley & Sons.

96. Wooldridge, M., y Ciancarini, P. (2001). Agent-oriented software engineering: The state of the art. En *Agent-Oriented Software Engineering*, Lecture Notes in Computer Science, 1957, 1-28. Springer-Verlag.

97. Yang, Q., Koutsopoulos, H., y Ben-Akiva, M. (2000). Simulation laboratory for evaluating dynamic traffic management systems. *Transportation Research Record: Journal of the Transportation Research Board*, 1710, 122-130.

Acrónimos

ABM - *Agent-Based Modeling*
AOSE - *Agent-Oriented Software Engineering*
BDI - *Beliefs-Desires-Intentions*
CDI - Creencias-Deseos-Intenciones
CMAC - *Cerebellar Model Articulation Controller*
CSIC - Consejo Superior de Investigaciones Científicas
CVE - Conductor-Vehículo-Entorno
DVE - *Driver-Vehicle-Environment*
EMF - *Eclipse Modeling Framework*
EMOF - *Essential MOF*
ES - *Environment Simulator*
GIS - *Geographic Information System*
GEF - *Graphical Editing Framework*
GPS - *Global Positioning System*
GRASIA - GRupo de Investigación en Aplicaciones Sociales e Interdisciplinares basadas en Agentes
HTML - *HyperText Markup Language*
IDK - *INGENIAS Development Kit*
IDM - *Intelligent Driver Model*
ISDM - Ingeniería del Software Dirigida por Modelos
ISOA - Ingeniería del Software Orientada a Agentes
LM - Lenguaje de Modelado
LMT - Lenguaje de Modelado de Tráfico
MBA - Modelado Basado en Agentes
MDE - *Model-Driven Engineering*
MOF - *Meta-Object Facility*
OCL - *Object Constraint Language*
OMG - *Object Management Group*
PUDS - Proceso Unificado de Desarrollo de Software
SDK - *Software Development Kit*
SMA - Sistema Multi-Agente
SPEM - *Software Process Engineering Metamodel*
SUMO – *Simulation of Urban Mobility*
TCP/IP - *Transmission Control Protocol / Internet Protocol*
TML - *Traffic Modelling Language*
UCM - Universidad Complutense de Madrid
UML - *Unified Modeling Language*

URI - *Uniform Resource Identifier*
USDP - *Unified Software Development Process*
XMI - *XML Metadata Interchange*
XML - *Extensible Markup Language*